

## 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. Класифікація за допомогою машин опорних векторів (SVM)

```
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, 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(',')

        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)

# 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]
```

					ДУ «Житомирська політехніка».22.121.07.806 – ІІЗк						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Кияшенко А.С.			Системи штучного інтелекту Лабораторна №2			Лім.	Арк.	Аркушів	
Перевір.									1	14	
Реценз.								ФІКТ Гр. ІІЗк-19-1			
Н. Контр.											
Затверд.											

```

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)

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]))
        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])

```

F1 score: 56.15%

Process finished with exit code 1

## Завдання 2. Порівняння якості класифікаторів SVM з нелінійними ядрами

### Поліноміальне ядро

```

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, cross_val_score

input_file = 'income data.txt'

```

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 10000

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)

# 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)

classifier = OneVsOneClassifier(SVC(random_state=0, 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(random_state=0, kernel='poly', degree=8))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

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])

```

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ППЗк	Арк.
		.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        else:
            input_data_encoded[i] =
int(label_encoder[count].transform(input_data[i]))
            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])

```

## Гаусове ядро

```

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, 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(',')

        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)

# 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)

classifier = OneVsOneClassifier(SVC(random_state=0, kernel='rbf'))

classifier.fit(X, y)

```

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = OneVsOneClassifier(SVC(random_state=0, kernel='rbf'))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

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[i].transform(input_data[i]))
        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])

```

F1 score: 71.95%

Process finished with exit code 1

## Сигмоїдальне ядро

```

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, 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(',')

```

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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)

# 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)

classifier = OneVsOneClassifier(SVC(random_state=0, 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.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

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[i].transform(input_data[i]))
        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])

```

```

C:\Users\Admin\AppData\Local\Microsoft\WindowsApps\python3.10.exe D:/LabsPoLi/AI/Lab2/LR_2_task_2_3.py
F1 score: 63.77%

```

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

### Завдання 3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

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

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
         'class']
dataset = read_csv('url', names=names)
# shape
print("*****shape*****")
print(dataset.shape)

print("*****Зріз даних*****")
print(dataset.head(20))

# description
print("*****Опис даних*****")
print(dataset.describe())

# classdistribution
print("*****Розподіл за класом*****")
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
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)

print("*****Оцінка моделей*****")
models = []
models.append(('LR', LogisticRegression(solver='liblinear',
multi_class='ovr')))
```

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
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()

model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

print("*****Оцінюємо прогноз*****")
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

from sklearn.neighbors import KNeighborsClassifier
import numpy as np

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train, Y_train)
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма масиву X_new: {}".format(X_new.shape))
prediction = knn.predict(X_new)
print("Прогноз: {}".format(prediction))
print("Спрогнозована метка: {}".format(dataset['class']))

y_pred = knn.predict(X_validation)
print("Прогнози для тестового набору:\n {}".format(y_pred))

print("Правильність тестового набору: {:.2f}".format(np.mean(y_pred ==
Y_validation)))

print("Правильність тестового набору: {:.2f}".format(knn.score(X_validation,
Y_validation)))

```

Форма масиву та зріз даних

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



```

*****shape*****
(150, 5)
*****Зріз даних*****
      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

```

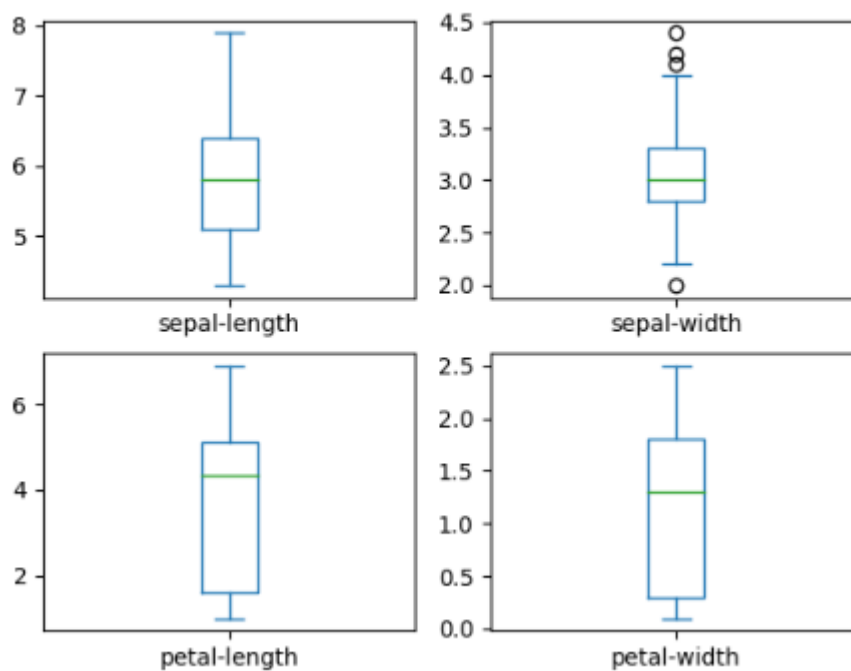
Статистичне зведення та розподіл за класом.

```

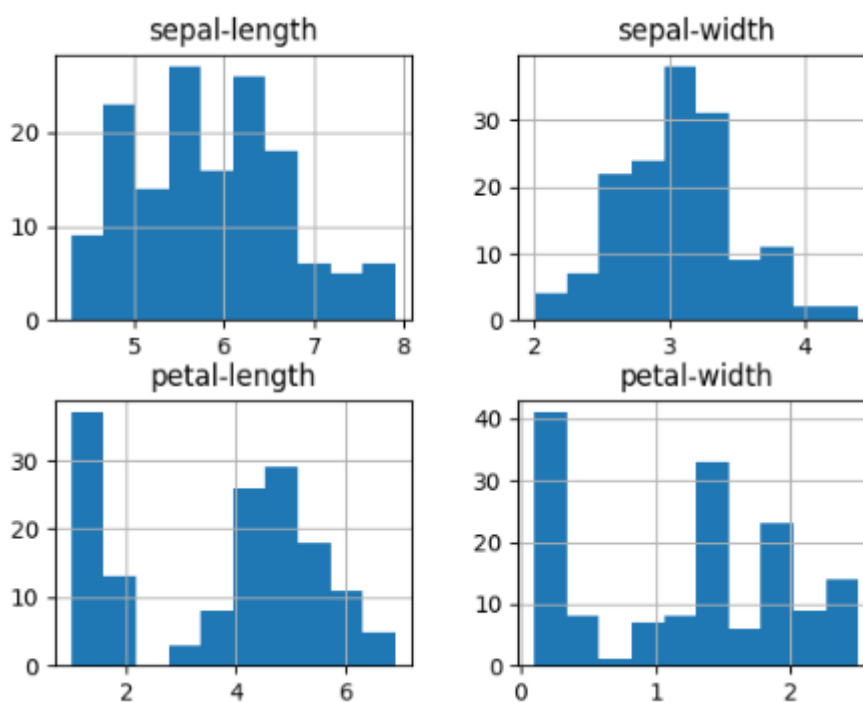
*****Опис даних*****
      sepal-length  sepal-width  petal-length  petal-width
count    150.000000    150.000000    150.000000    150.000000
mean       5.843333     3.054000     3.758667     1.198667
std        0.828066     0.433594     1.764420     0.763161
min        4.300000     2.000000     1.000000     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
*****Розподіл за класом*****
class
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64

```

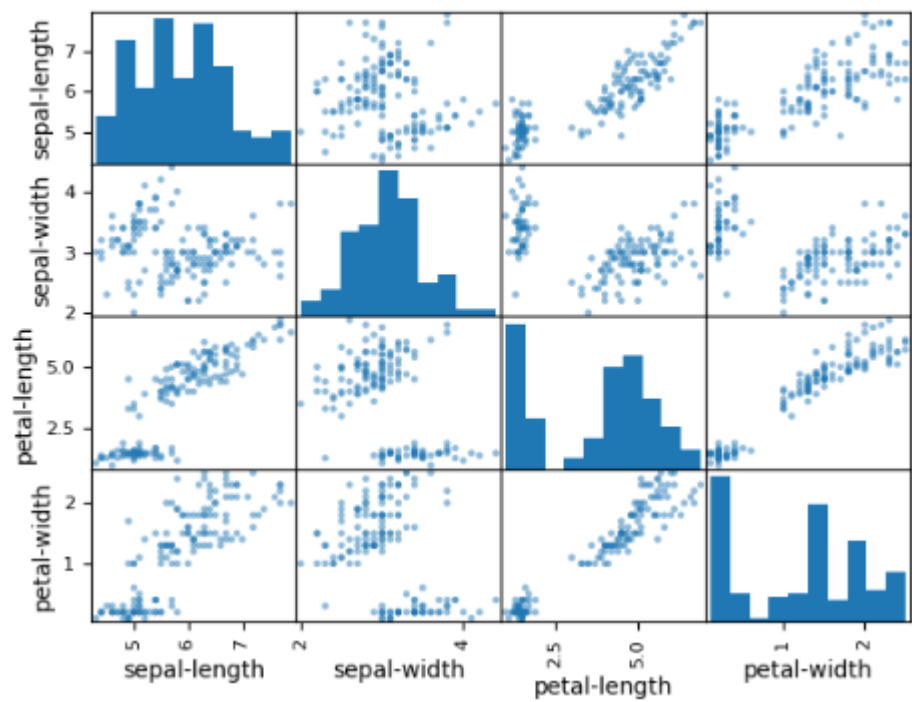
## Діаграма розмаху



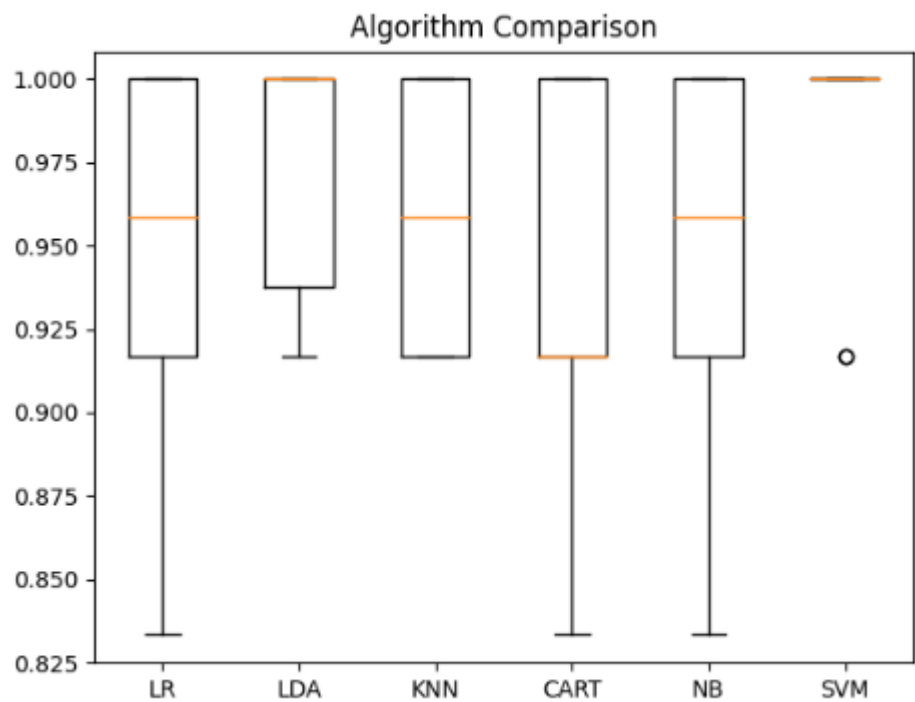
## Гістограма розподілу атрибутів датасета



Матриця діаграм розсіювання



Порівняння алгоритмів



```

*****Оцінка моделей*****
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.053359)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)

```

Найкращий результат має алгоритм SVM.

Мій прогноз

```

Форма масиву X_new: (1, 4)
Прогноз: ['Iris-setosa']
Спрогнозована метка: 0      Iris-setosa

```

Квітка належить до класу Iris-setosa

#### Завдання 4. Класифікація даних лінійним класифікатором Ridge

```

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,
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 Corcoef:',
      np.round(metrics.matthews_corrcoef(ytest, ypred), 4))
print('\t\tClassification Report:\n',
      metrics.classification_report(ypred, ytest))

sns.set()

mat = confusion_matrix(ytest, ypred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')

```

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

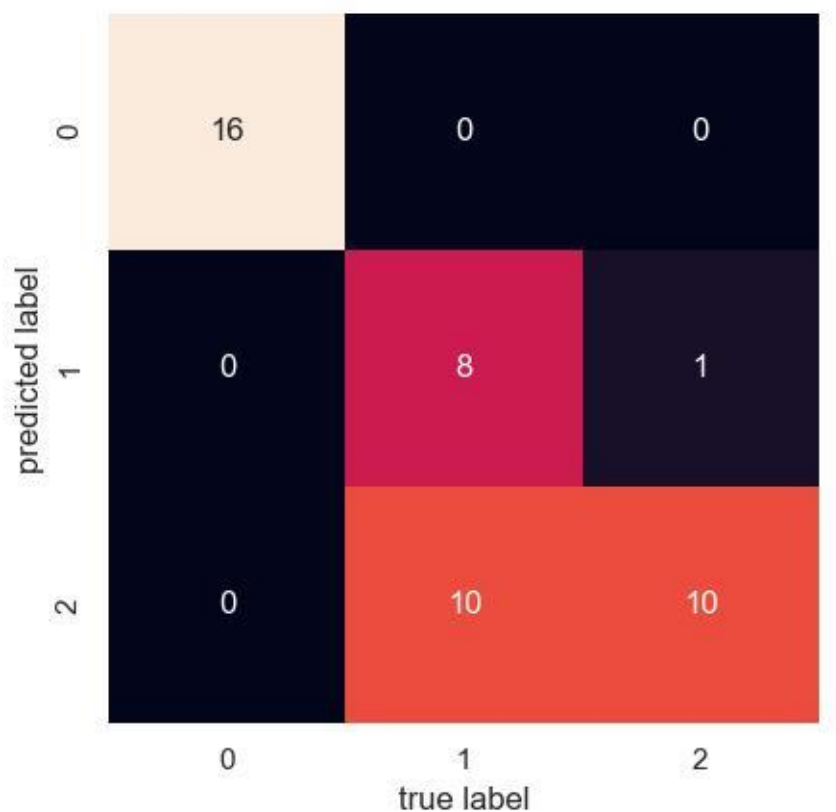
```
plt.ylabel('predicted label')
plt.savefig("Confusion.jpg")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format="svg")
```

```
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrccoef: 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        0.76        0.76        45
 macro avg          0.78        0.80        0.75        45
weighted avg          0.85        0.76        0.76        45
```



Ця картинка показує, що програма прогнозує 16 із 16 класів 0, 8 з 18 класів 1 та 10 з 11 класів 2.

Посилання на Git: <https://github.com/Grum74/AI>

### **Висновок**

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

		Кияшенко А.С.			ДУ «Житомирська політехніка».22.121.07. 806– ПЗк	Арк.
		.				14
Змн.	Арк.	№ докум.	Підпис	Дата		