

# 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 2.1.

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import 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)
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)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X=X, y=Y)
```

					ДУ «Житомирська політехніка».22.121.08.000 – Лр2			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Коптяєв М.П.			Звіт з лабораторної роботи		Лім.	Арк.
Перевір.		Філіпов В.О.						1
Керівник								12
Н. контр.							ФІКТ Гр. ІПЗк-20-1[1]	
Зав. каф.								

```

X_train, X_test, y_train, y_test \
= train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted',
cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
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 = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])

```

```

Accuracy: 81.95%
Precision: 80.94%
Recall: 81.95%
F1: 80.13%
F1 score: 80.13%
>50K

```

Рис. 1 - Результат виконання

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

Завдання 2.2.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

```
Accuracy 83.99%  
Precision: 83.21%  
Recall: 83.99%  
F1: 82.99%  
F1 score: 82.99%  
<=50K
```

Рис. 2 Поліноміальне ядро

```
Accuracy: 83.96%  
Precision: 83.18%  
Recall: 83.96%  
F1: 82.95%  
F1 score: 82.95%  
<=50K
```

Рис. 3 Гаусове ядро

```
Accuracy: 57.26%  
Precision: 57.1%  
Recall: 57.26%  
F1: 57.18%  
F1 score: 57.18%  
<=50K
```

Рис. 4 Сигмоїдальне ядро

RFB дало найкращий результат. Сигмоїдальне ж відстає по швидкості та точності.

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філіпов В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

## Завдання 2.3.

```
# -*- coding: utf-8 -*-
from sklearn.datasets import load_iris
import numpy as np
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
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("Назви 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']))
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)
# shape
print(dataset.shape)
# Зріз даних head
print(dataset.head(20))
# Статистичні зведення методом describe
print(dataset.describe())
# Розподіл за атрибутом class
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)
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
```

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Пр2	Арк.
		Філіпов В.О.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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, 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]])
for name, model in models:
    model.fit(X_train, Y_train)
    prediction = model.predict(X_new)
    print("Прогноз: {}".format(prediction))
    print(accuracy_score(Y_validation, predictions))
    print(confusion_matrix(Y_validation, predictions))
    print(classification_report(Y_validation, predictions))

```

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

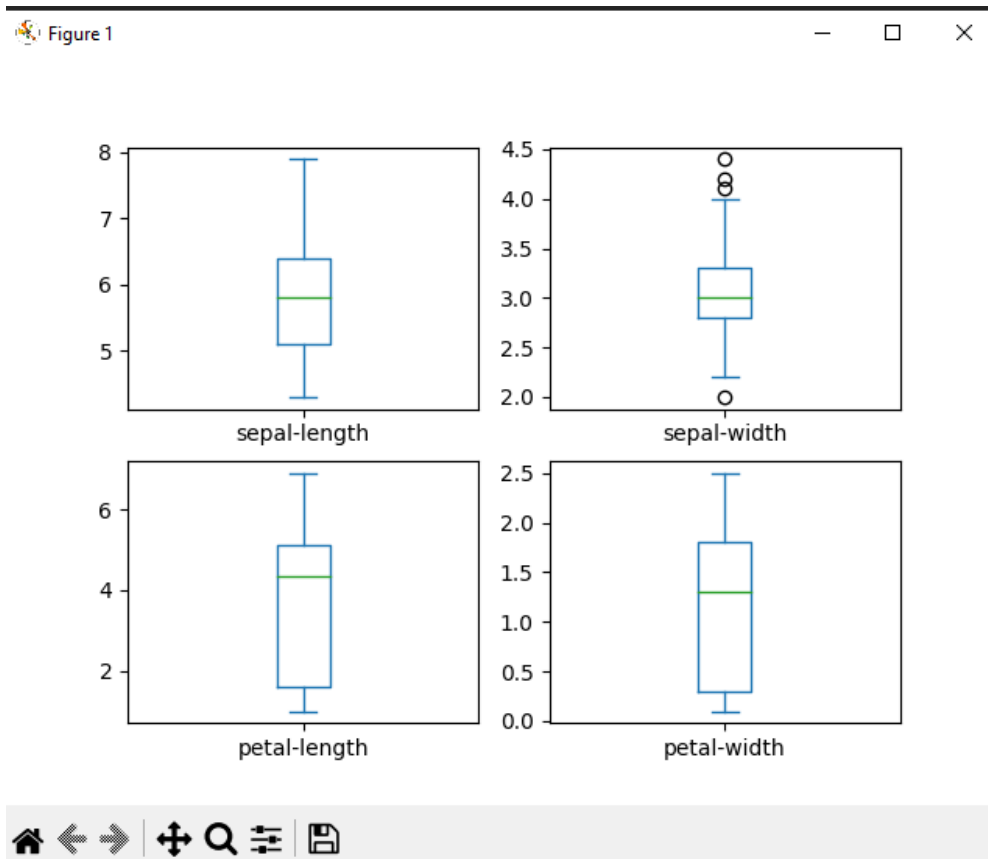


Рис. 5 - Діаграма розмаху

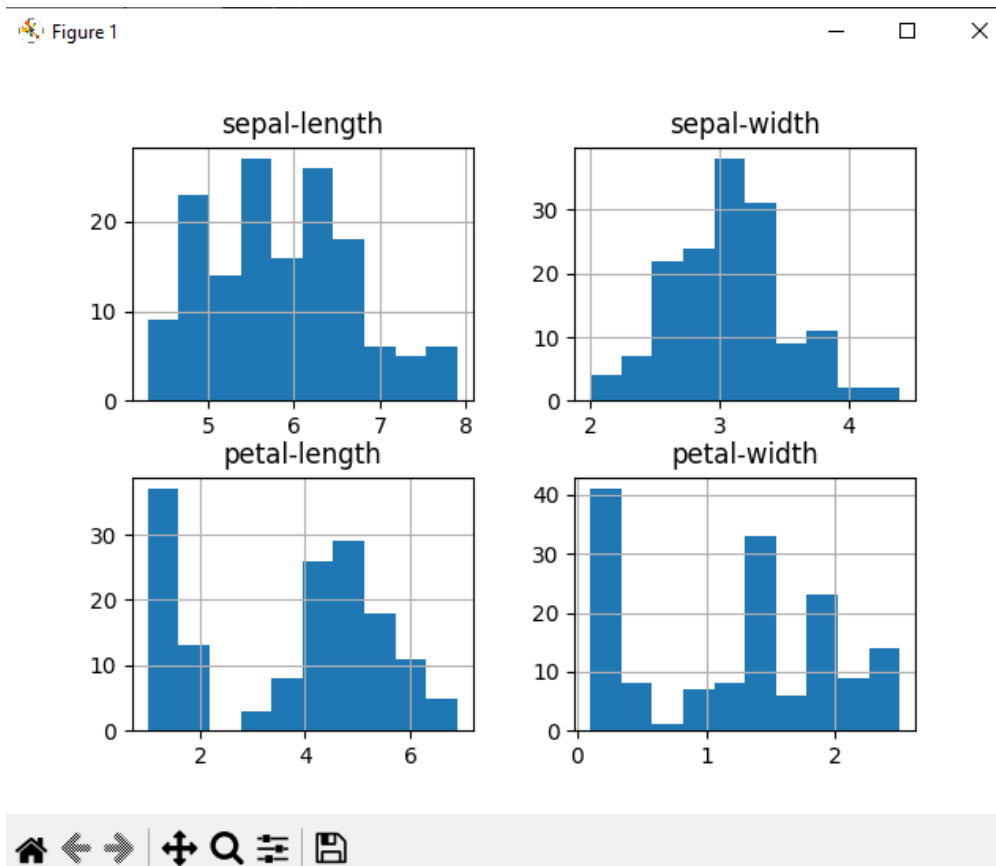


Рис. 6 - Діаграми розподілу атрибутів

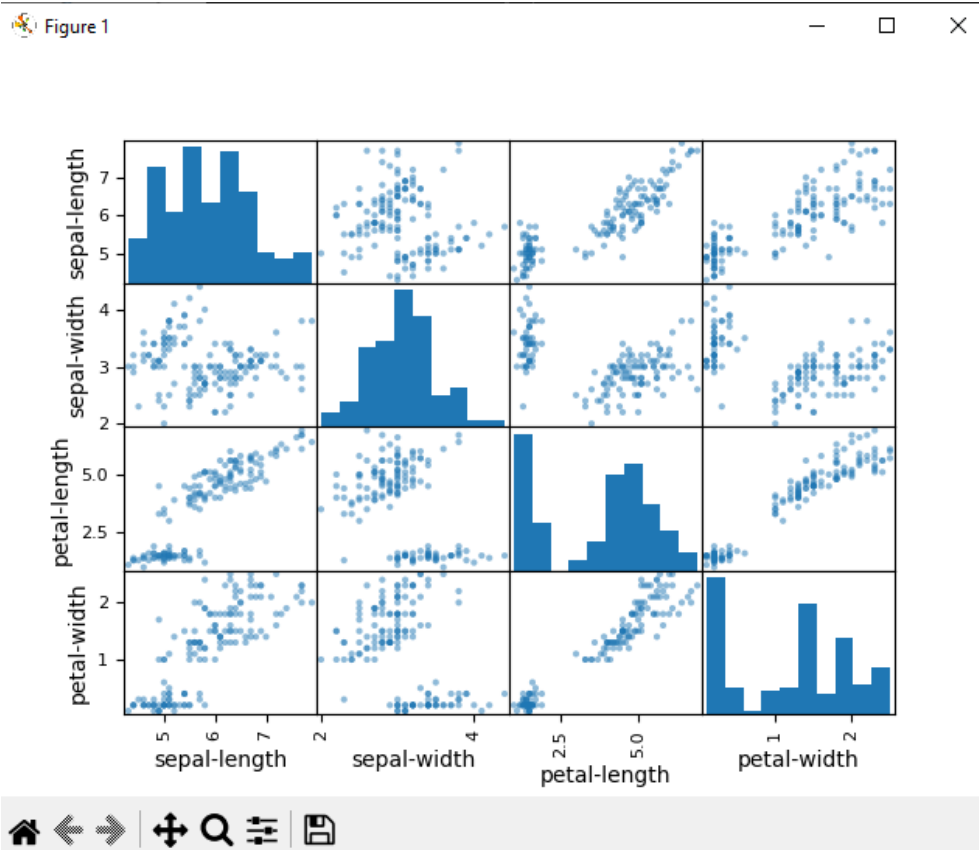


Рис. 7 - Діаграми розсіювання

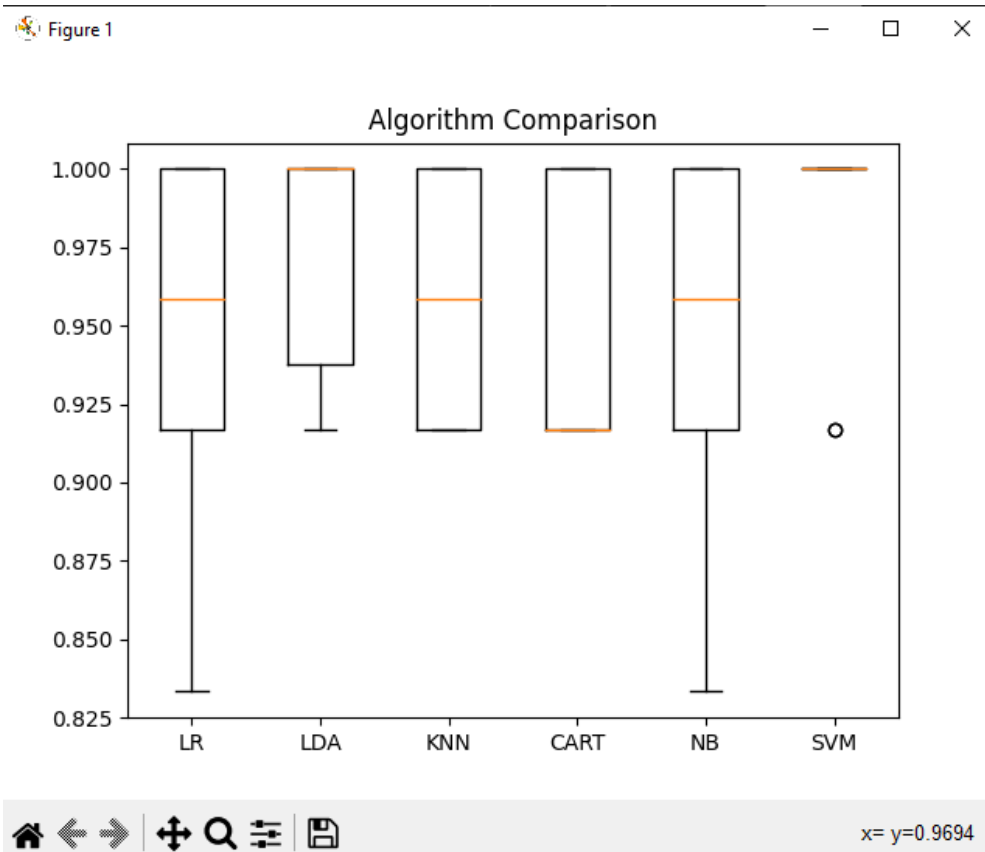


Рис. 8 - Порівняння алгоритмів

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Пр2	Арк.
		Філіпов В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		





## Завдання 2.4.

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_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

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)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)

#classifier = LogisticRegression(solver='liblinear', multi_class='ovr')

#classifier = LogisticRegression(solver='liblinear', multi_class='ovr')
#classifier = LinearDiscriminantAnalysis()
#classifier = KNeighborsClassifier()
#classifier = DecisionTreeClassifier()
#classifier = GaussianNB()
classifier = SVC(gamma='auto')

classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test \
    = train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
```

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філіпов В.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(
    classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
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 = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])

```

```

Accuracy: 81.14%
Precision 79.86%
Recall: 81.14%
F1: 79.35%
F1 score: 79.35%
>50K

```

Рис. 10 Точність класифікатора LDA

```

Accuracy: 82.16%
Precision 81.53%
Recall: 82.16%
F1: 81.75%
F1 score: 81.75%
<=50K

```

Рис. 11 Точність класифікатора KNN

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філінов В.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Accuracy: 80.55%
Precision 80.76%
Recall: 80.66%
F1: 80.84%
F1 score: 80.77%
>50K

```

Рис. 12 Точність класифікатора CART

```

Accuracy: 79.76%
Precision 78.2%
Recall: 79.76%
F1: 77.13%
F1 score: 77.13%
<=50K

```

Рис. 13 Точність класифікатора NB

```

Accuracy: 82.38%
Precision 81.51%
Recall: 82.38%
F1: 80.6%
F1 score: 80.6%
>50K

```

Рис. 14 Точність класифікатора SVM

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філіпов В.О.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

## Завдання 2.5.

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

sns.set()
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 Corrcoeff:', 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")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format="svg")
```

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філіпов В.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 14 - Результат виконання

Каппа – це стат. значення, що вимірює міжрегіональну згоду на категоріальні предмети і вважається більш надійнішим аніж розрахунок у відсотках. Також було отримано коеф. кореляції Метьюза – використовується в машинному навчанні, як міра якості бінарних мультикласних класифікацій.

Матриця невідповідності – це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму, зазвичай керованого навчання. Кожен з рядків цієї матриці представляє зразки прогнозованого класу, тоді як кожен зі стовпців представляє зразки справжнього класу.

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

Git: <https://github.com/Drakoshik/AI.git>

		Коптяєв М.П.			ДУ «Житомирська політехніка».22.121.08.000 – Лр2	Арк.
		Філінов В.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		