

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

<https://github.com/MaximVengel/AI>

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

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 containing data
input_file = 'income_data.txt'

# Read the data
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

# Convert to numpy array
X = np.array(X)
```

					ДУ «Житомирська політехніка».23.121.06.000 – Лр2			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Венгель М.І.			Звіт з лабораторної роботи		Лім.	Арк.
Перевір.		Голенко М.Ю.						1
Керівник							Аркушів	
Н. контр.							16	
Зав. каф.							ФІКТ Гр. ІПЗ-20-2	

```

# Convert string data to numerical data
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)

# Create SVM classifier
classifier = OneVsOneClassifier(LinearSVC(random_state=0))

# Train the classifier
classifier.fit(X=X, y=y)

# Cross validation
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))

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)

# Compute the F1 score of the SVM classifier
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)) + "%")

# Predict output for a test datapoint
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-
States']

# Encode test datapoint
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

```

		Венгелъ М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				
Змн.	Арк.	№ докум.	Підпис	Дата		2

```

input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]

# Run classifier on encoded datapoint and print output
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])

```

Результат виконання:

```

Run: LR_2_task_1 x
C:\Users\38098\Desktop\lab02\venv\Scripts\python.exe
Accuracy: 82.01%
Precision: 80.96%
Recall: 82.01%
F1: 80.1%
F1 score: 80.1%
>50K

Process finished with exit code 0

```

Рис. 1. Завдання №2.1

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами.

Результат виконання:

```

Accuracy: 83.5%
Precision: 82.84%
Recall: 83.5%
F1: 83.01%
F1 score: 83.01%
<=50K

```

Рис. 2. Завдання №2.2 (Poly-ядро)

```

LR_2_task_1 x
C:\Users\38098\Desktop\lab02\venv
Accuracy: 82.01%
Precision: 80.96%
Recall: 82.01%
F1: 80.1%
F1 score: 80.1%
>50K

```

Рис. 3. Завдання №2.2 (gamma-ядро)

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_2_task_1 x
C:\Users\38098\Desktop\lab02\venv\Scripts\pyt
Accuracy: 58.2%
Precision: 57.85%
Recall: 58.2%
F1: 58.02%
F1 score: 58.02%
<=50K

```

Рис. 4. Завдання №2.2 (sigmoid-ядро)

Висновки до завдання: RFB дає доволі точний результат, але має недолік перед poly-ядром, зате має перевагу в швидкодії. Сигмоїдне ядро дає більш низькі результати у точності та швидкодії. З цього можемо вважати, що для нашого випадку, краще підходить RBF, тому що має поєднання точності та швидкості водночас.

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

LR_2_task_3.py

```

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(f'Ключі iris_dataset: {iris_dataset.keys()}')
print(iris_dataset['DESCR'][:193] + "\n....")
print(f"Назви відповідей: {iris_dataset['target_names']}")
print(f"Назва ознак: {iris_dataset['feature_names']}")
print(f"Тип масиву data: {type(iris_dataset['data'])}")
print(f"Форма масиву data: {iris_dataset['data'].shape}")
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()))
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)

```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Пр2	Арк.
		Голенко М.Ю.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Оцениваем 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))
```

Результат виконання:

[illegible]

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_2_task_3 x
11      4.8      3.4      1.6      0.2 Iris-setosa
12      4.8      3.0      1.4      0.1 Iris-setosa
13      4.3      3.0      1.1      0.1 Iris-setosa
14      5.8      4.0      1.2      0.2 Iris-setosa
15      5.7      4.4      1.5      0.4 Iris-setosa
16      5.4      3.9      1.3      0.4 Iris-setosa
17      5.1      3.5      1.4      0.3 Iris-setosa
18      5.7      3.8      1.7      0.3 Iris-setosa
19      5.1      3.8      1.5      0.3 Iris-setosa

      sepal-length  sepal-width  petal-length  petal-width
count    150.000000    150.000000    150.000000    150.000000
mean       5.843333      3.054000      3.758667      1.198667
std        0.828066      0.433594      1.764420      0.763161
min        4.300000      2.000000      1.000000      0.100000
25%        5.100000      2.800000      1.600000      0.300000
50%        5.800000      3.000000      4.350000      1.300000
75%        6.400000      3.300000      5.100000      1.800000
max        7.900000      4.400000      6.900000      2.500000

class
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64

```

```

LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.038188)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]

      precision    recall  f1-score   support

```

Рис. 5. Вивід у консолі.

Figure 1

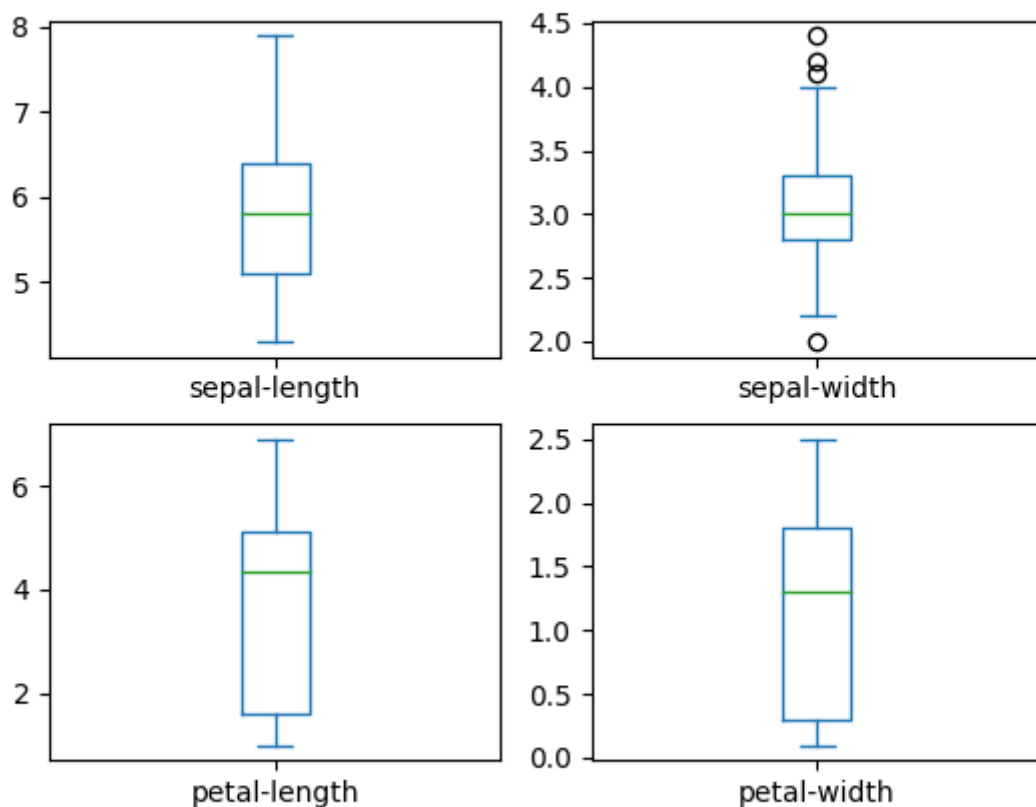
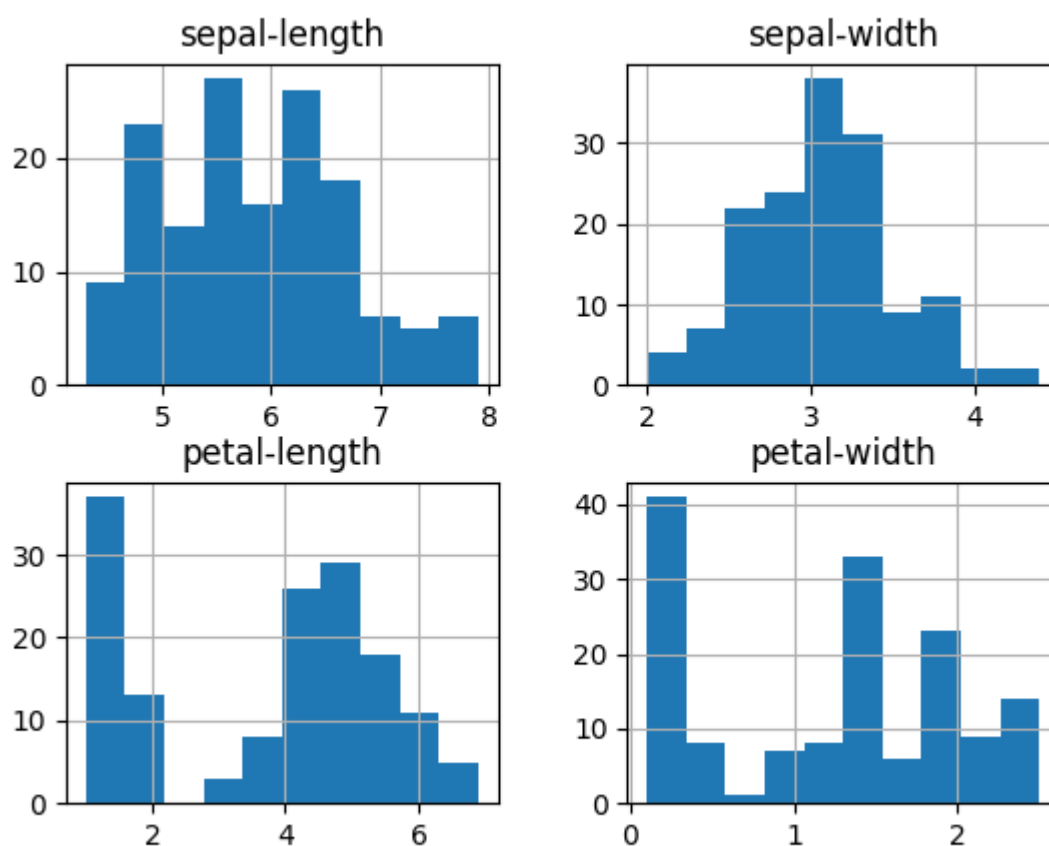


Рис. 6. Результат діаграми розмаху.

Figure 1



x=5.86 y=13.6

Рис. 7. Гістограма розподілу атрибутів.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

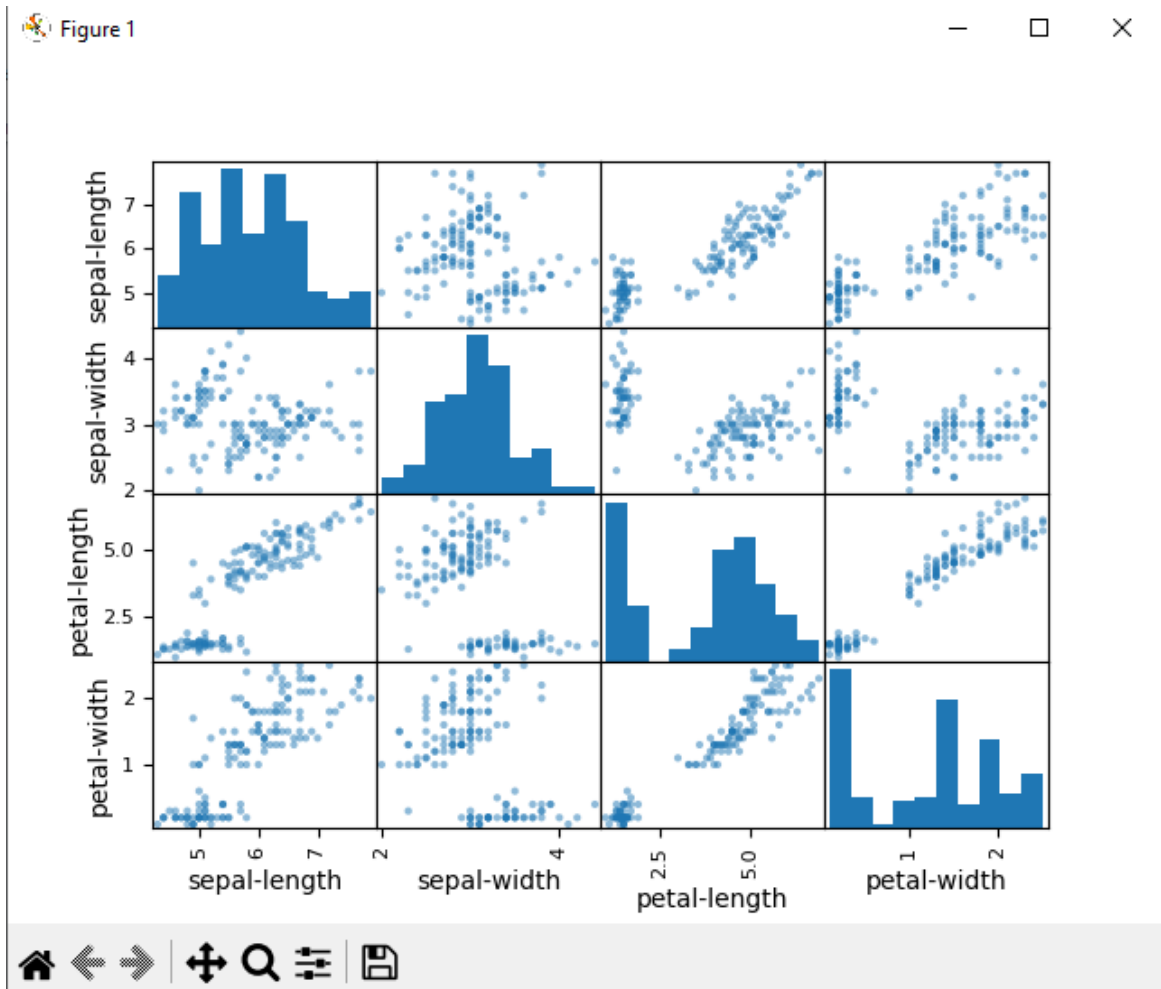


Рис. 8. Матриця діаграми розсіювання.

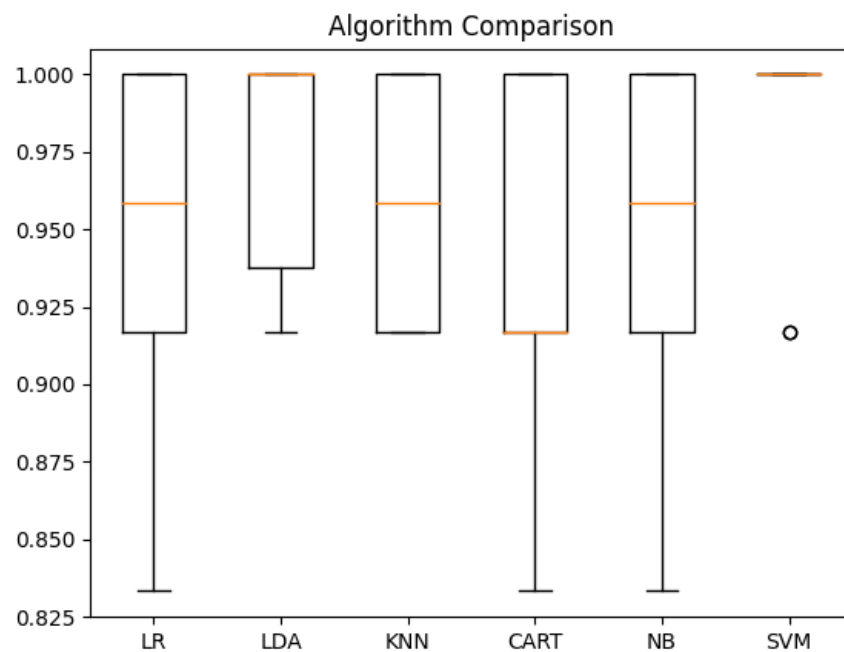


Рис. 9. Порівняння алгоритмів.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновок до завдання: квітка належала до класу **Iris-Setosa**. З діаграм робимо висновок, що найкраще показала себе модель лінійного дискримінантного аналізу, але вона є не дуже стійка і під час тестування займала більше часу для виконання.

Завдання 2.4 Порівняння якості класифікаторів для набору даних завдання 2.1.

LR_2_task_4.py

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

# Input file containing data
input_file = 'income_data.txt'

# Read the data
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

# Convert to numpy array
X = np.array(X)

# Convert string data to numerical data
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])
```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

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

# Create SVM classifier
classifier = GaussianNB()

# Train the classifier
classifier.fit(X=X, y=y)

# Cross validation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = GaussianNB()

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)

# Compute the F1 score of the SVM classifier
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)) + "%")

# Predict output for a test datapoint
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-
States']

# Encode test datapoint
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]

# Run classifier on encoded datapoint and print output
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])

```

Результат виконання:

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Accuracy: 81.85%  
Precision: 80.68%  
Recall: 81.85%  
F1: 80.13%  
F1 score: 80.13%  
>50K
```

Рис. 10. Точність класифікатора LR.

```
Accuracy: 81.35%  
Precision: 80.04%  
Recall: 81.35%  
F1: 79.51%  
F1 score: 79.51%  
>50K
```

Рис. 11. Точність класифікатора LDA.

```
Accuracy: 82.43%  
Precision: 81.79%  
Recall: 82.43%  
F1: 82.01%  
F1 score: 82.01%  
<=50K
```

Рис. 12. Точність класифікатора KNN.

```
Accuracy: 80.49%  
Precision: 80.91%  
Recall: 80.66%  
F1: 80.83%  
F1 score: 80.96%  
<=50K
```

Рис. 13. Точність класифікатора CART.

```

Accuracy: 82.3%
Precision: 81.47%
Recall: 82.3%
F1: 80.26%
F1 score: 80.26%
<=50K

```

Рис. 14. Точність класифікатора SVM.

```

Accuracy: 80.1%
Precision: 78.51%
Recall: 80.1%
F1: 77.53%
F1 score: 77.53%
<=50K

```

Рис. 15. Точність класифікатора NB.

Висновок до завдання: точність класифікатора KNN – найбільша.

Завдання 2.5 Класифікація даних лінійним класифікатором Ridge.

LR_2_task_5.py

```

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 Corrcoef:', np.round(metrics.matthews_corrcoef(ytest, ypred), 4))

```

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

```
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")
```

Результат виконання:

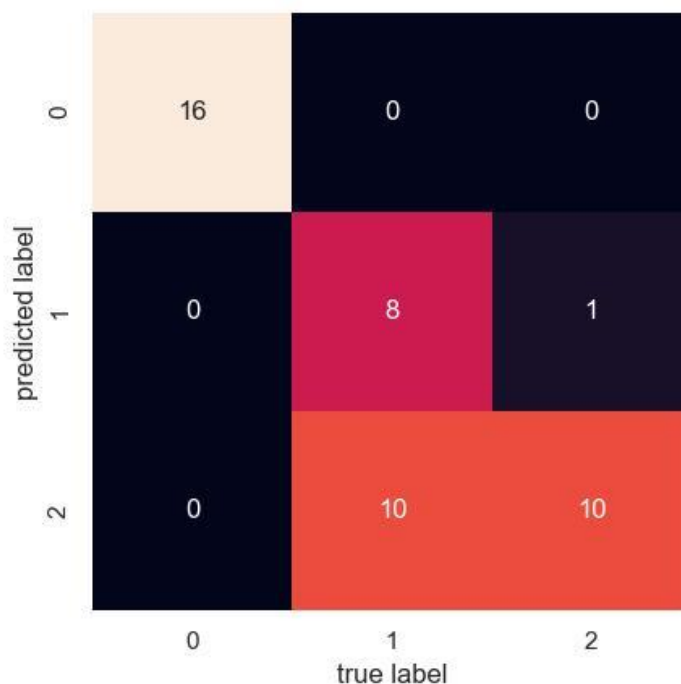
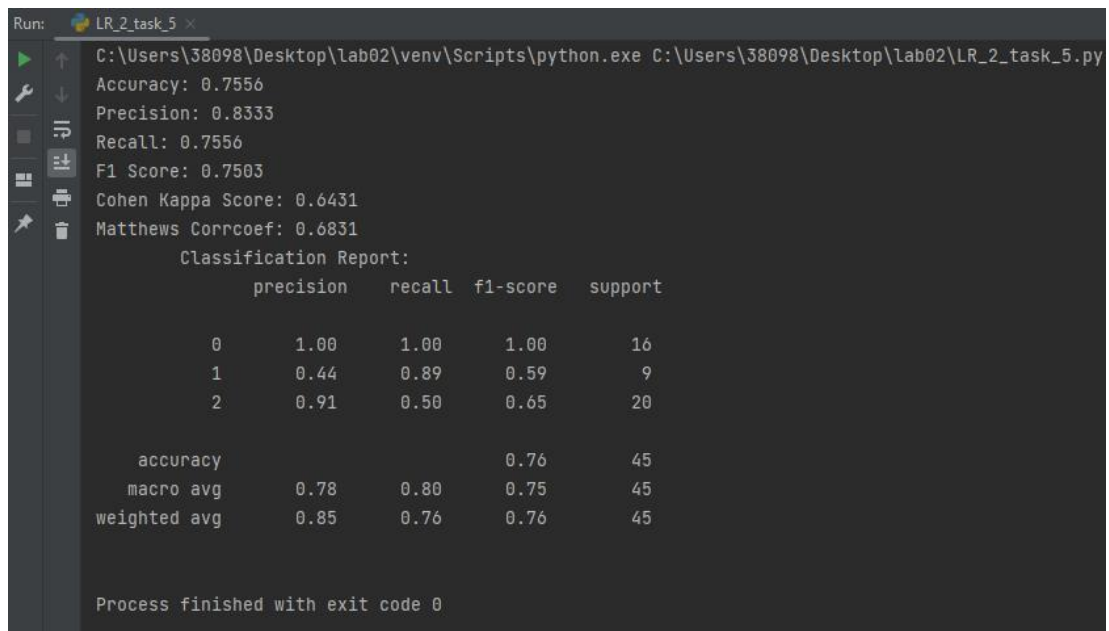


Рис. 16. Результат виконання завдання.

		Венгел М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновок до завдання: було отримано $r1$, recall, коефіцієнт Коена 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 (або навпаки). Її назва походить від того факту, що вона дає можливість просто бачити, чи допускає система невідповідності між цими двома класами.

Висновок по лабораторній роботі:

Використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.

		Венгель М.І.			ДУ «Житомирська політехніка».23.121.06.000 – Лр2	Арк.
		Голенко М.Ю.				16
Змн.	Арк.	№ докум.	Підпис	Дата		