Лабораторна робота №1

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

Посилання на гіт: https://github.com/IPZ213mmv/Lab2

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

Лістинг програми:

```
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
from sklearn.model_selection import cross_val_score
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])
X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

# Create SVM classifier
classifier = OneVsOneClassifier(LinearSVC(random_state=0, dual=False))

# Train the classifier
classifier.fit(X, 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, dual=False))
classifier.fit(X_train, 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)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.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',
'UnitedStates']

# Encode test datapoint
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).reshape(1, -1)
# 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:

```
F1: 76.01%
Акуратність: 79.66%
Повнота: 79.66%
Точність: 78.88%
```

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

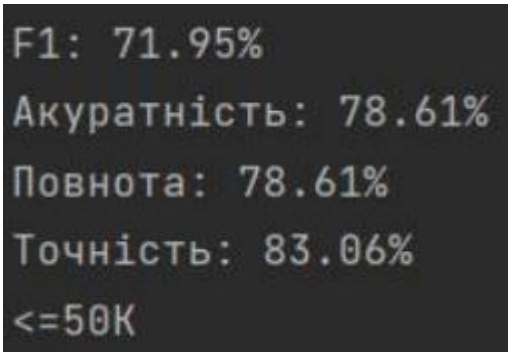
Лістинг програми:

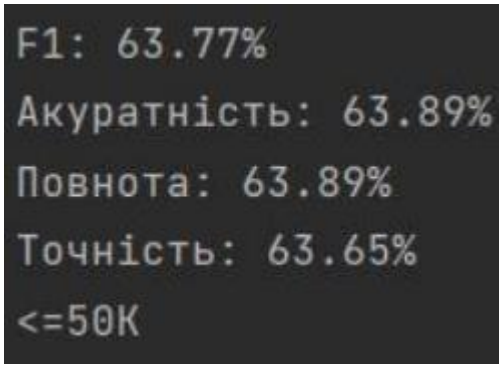
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
# Input file containing data
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])

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

# Create SVM classifier
classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))  # kernel='poly'
# kernel='rbf'

# Train the classifier
classifier.fit(X, 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(SVC(kernel='sigmoid'))  # kernel='poly'
# kernel='rbf'
classifier.fit(X_train, 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)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.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',
```

```
'UnitedStates']

# Encode test datapoint
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).reshape(1, -1)
# 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:



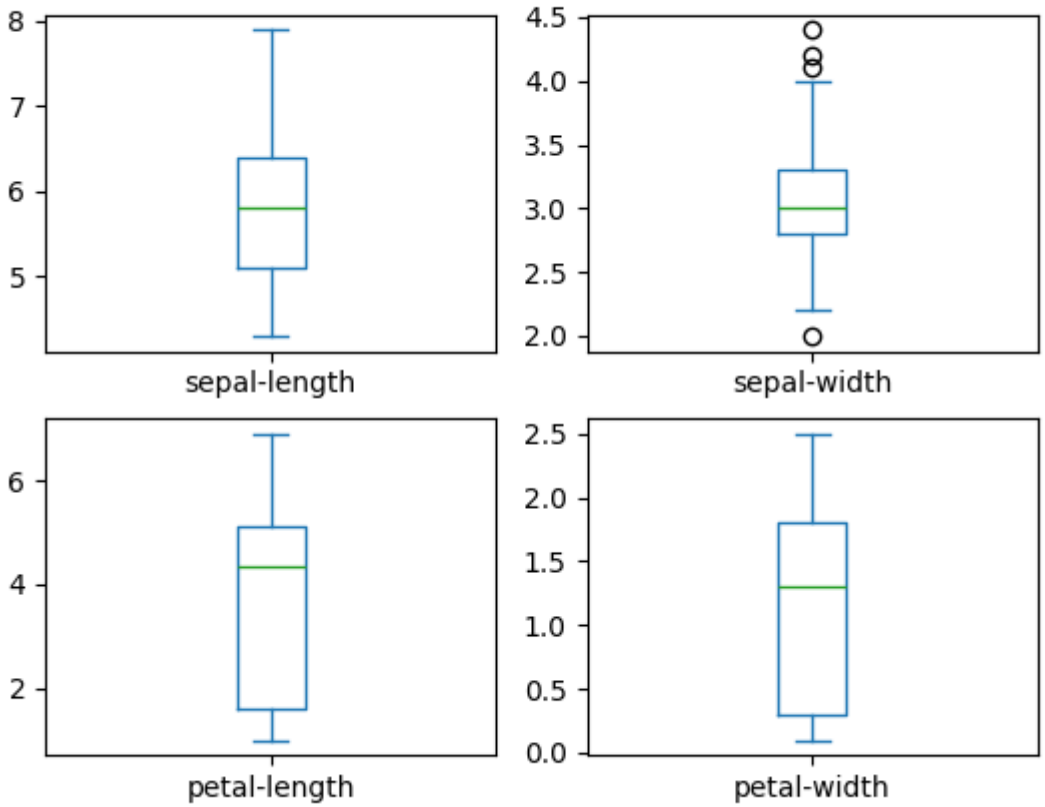


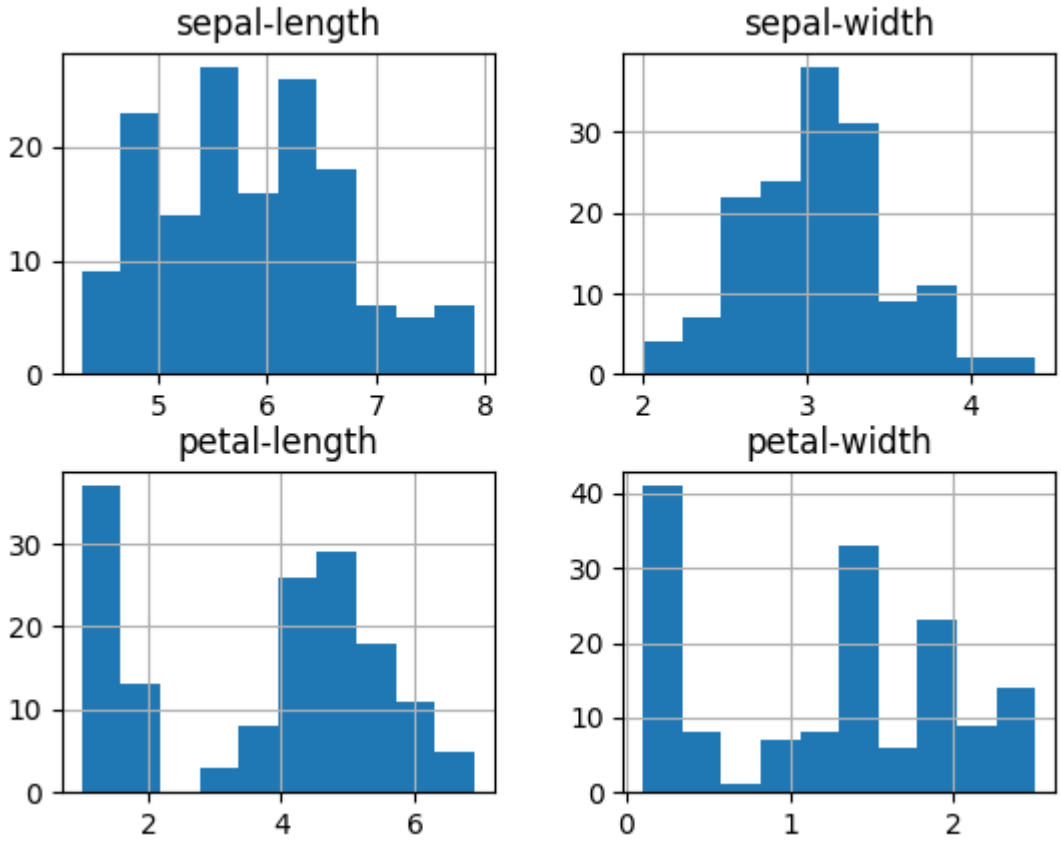
Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

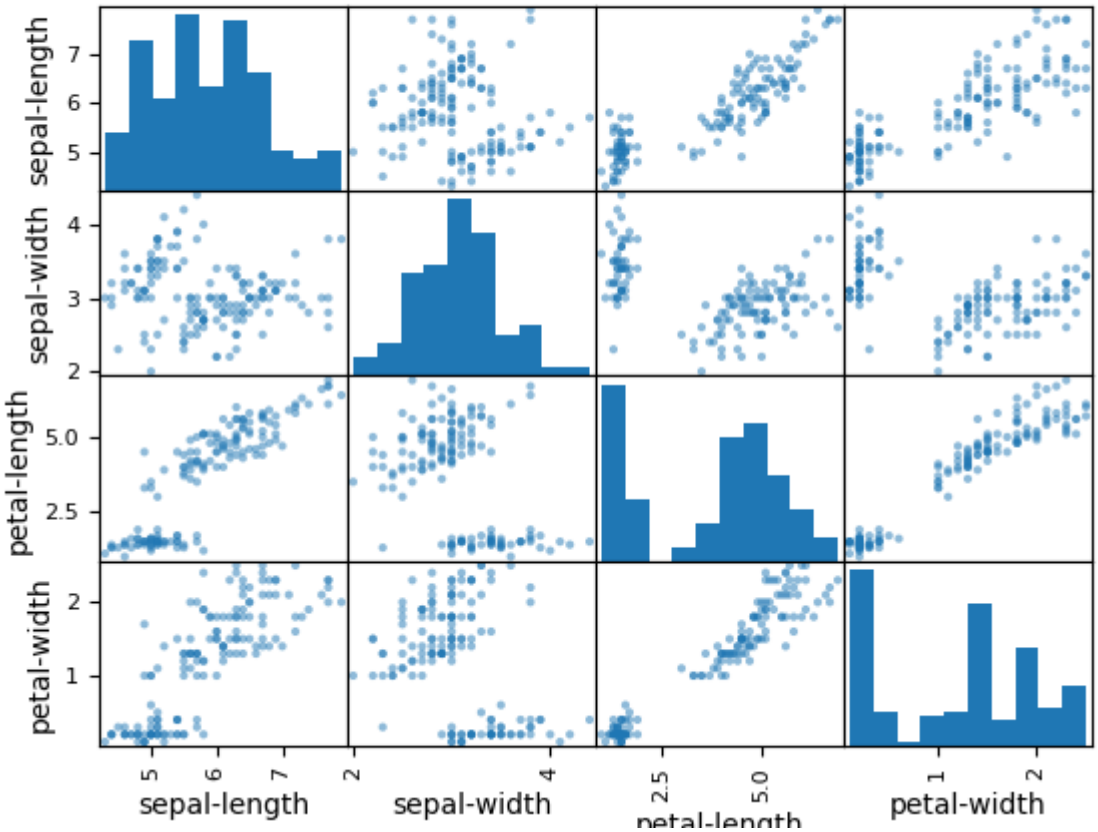
Лістинг програми:

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
# Load dataset
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)
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
pyplot.show()
# histograms
dataset.hist()
pyplot.show()
# scatter plot matrix
```

```
scatter_matrix(dataset)
pyplot.show()
```




sepal-length
sepal-width
petal-length
petal-width


sepal-length
sepal-width
petal-length
petal-width

Результат:

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг програми:

```
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
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

# 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])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)
# Cross validation
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,random_state=5)

classifier = OneVsOneClassifier(LinearRegression())
classifier.fit(X_train, y_train)
```

```
y_test_pred = classifier.predict(X_test)
print("")
print("LR:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(LinearDiscriminantAnalysis())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("LDA:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(KNeighborsClassifier())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("KNN:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(DecisionTreeClassifier())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("CART:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")

classifier = OneVsOneClassifier(SVC())
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print("")
print("SVM:")
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100*f1.mean(), 2)) + "%")
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Акуратність: " + str(round(100*accuracy.mean(), 2)) + "%")
```

```
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)
print("Повнота: " + str(round(100*recall.mean(), 2)) + "%")
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
print("Точність: " + str(round(100*precision.mean(), 2)) + "%")

X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5], [6, 5], [5.6, 5], [3.3,
0.4], [3.9, 0.9], [2.8, 1], [0.5, 3.4], [1, 4], [0.6, 4.9]])
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])

# Create the logistic regression classifier
classifier = linear_model.LogisticRegression(solver='liblinear', C=1)
# classifier = linear_model.LogisticRegression(solver='liblinear', C=100)

# Train the classifier
classifier.fit(X, y)
# Visualize the performance of the classifier
visualize_classifier(classifier, X, y)
```

```
LR:
F1: 31.36%
Акуратність: 36.61%
Повнота: 36.61%
Точність: 81.24%

LDA:
F1: 79.35%
Акуратність: 81.14%
Повнота: 81.14%
Точність: 79.86%

KNN:
F1: 48.04%
Акуратність: 46.48%
Повнота: 46.48%
Точність: 70.48%

CART:
F1: 80.69%
Акуратність: 80.57%
Повнота: 80.7%
Точність: 80.75%

SVM:
F1: 71.95%
Акуратність: 78.61%
Повнота: 78.61%
Точність: 83.06%
```

Результат:

Висновок: на цій лабораторній роботі я, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідив різні методи класифікації даних та навчився їх порівнювати