Лабораторна робота №2
Тема: Порівняння методів класифікації даних

Завдання 1: Класифікація за допомогою машин опорних векторів SVM
Посилання на гіт хаб: https://github.com/ViMIMercurysMight/python-ai.git

На початку роботи випишимо атрибути що надаються з набору даних та визначимо їх тип (таб. 1).

Таблиця 1. Опис атрибутів

Атрибут	Опис	Тип
Вік	Вік	Числова
Тип зайнятості	Тип зайнятості (ФОП, працює на невону ставку тощо)	Категоріальна
Вага	Коефіцієнт вани у файлах поточного опитування населення	Числова
Освіта	Рівень освіти (бакалавр, неповне середне тощо)	Категоріальна
Освітній номер	Номер рівня освіти	Числова
Подружній стан	Одруженний, розлученний тощо (поточний статус)	Категоріальна
Професійна діяльність	Рід зайнять	Категоріальна
Сімейний стан	Має дитину, розлученний тошо	Категоріальна
Раса	Расова приналежність	Категоріальна
Стать	Статева приналежність	Категоріальна
Кріїна походження	Країна проживання (поточна)	Категоріальна
Дохід	Середній дохід в рік	Категорівльна
Годин в неденю працює	Кількість робочих годин в неділю	Числова
Капітал зростання	Рівень росту доходу	Числова

Після описання атрибутів напишимо программу для опису значення показника якості F1 за зразком доповнивши програму пошуком інших показників якості. Обрана точка позначення доходу не дивлячись на наявність чисел в нашому випадку скоріше виступає в якості категоріальної (визначає групи чий дохід вище або нижче за деяке значення)

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
import sklearn.svm
from sklearn import preprocessing
from sklearn.svm import LinearSVC
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.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score


#Input Data
input_file = "income_data.txt"

X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 500

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)

print("---------[X After file reading]--------\n")
print(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)

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', 'Han-
dlers-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 ]

print("-_-_-_-_-_-_-")
print(input_data_encoded)

predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])
```

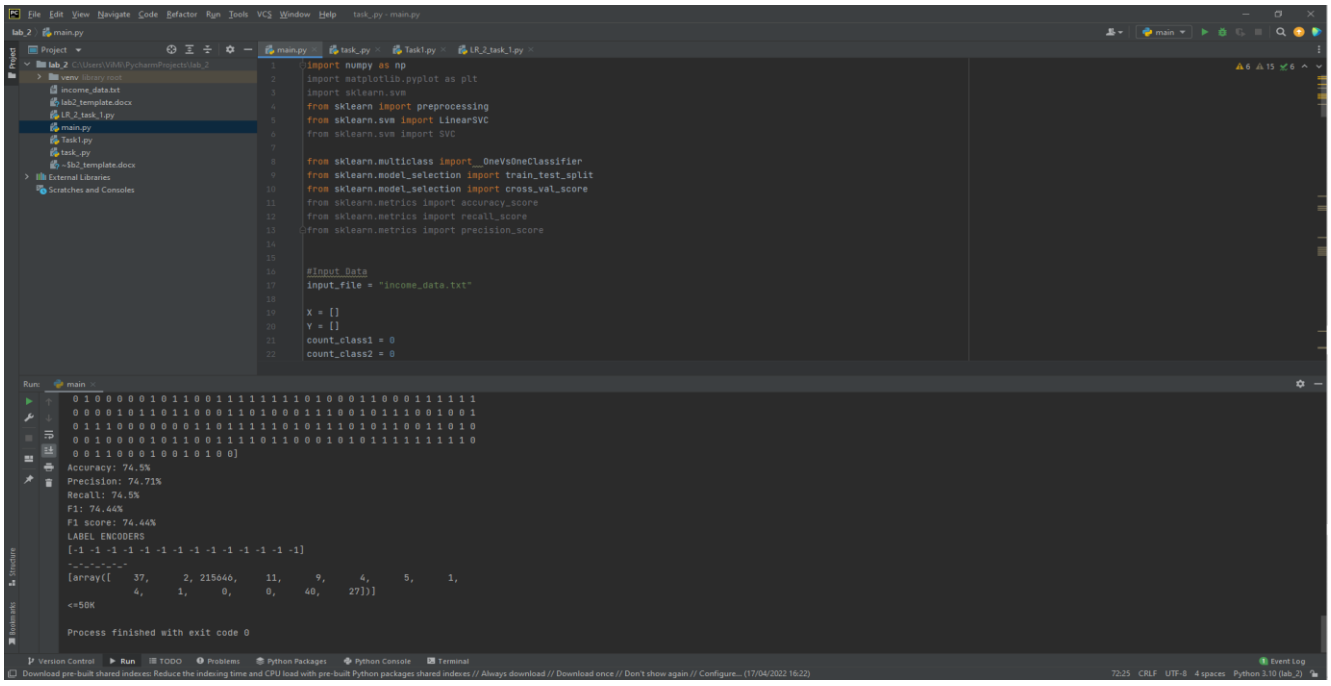

Рисунок 1. Результати роботи программи

Завдання 2: Порівняння якості класифікаторів SVM з нелінійними ядрами

Наступним кроком обчислимо значення показників з використанням різних ядер класифікатора. Коди програм наведено окремими файлами в репозиторії.

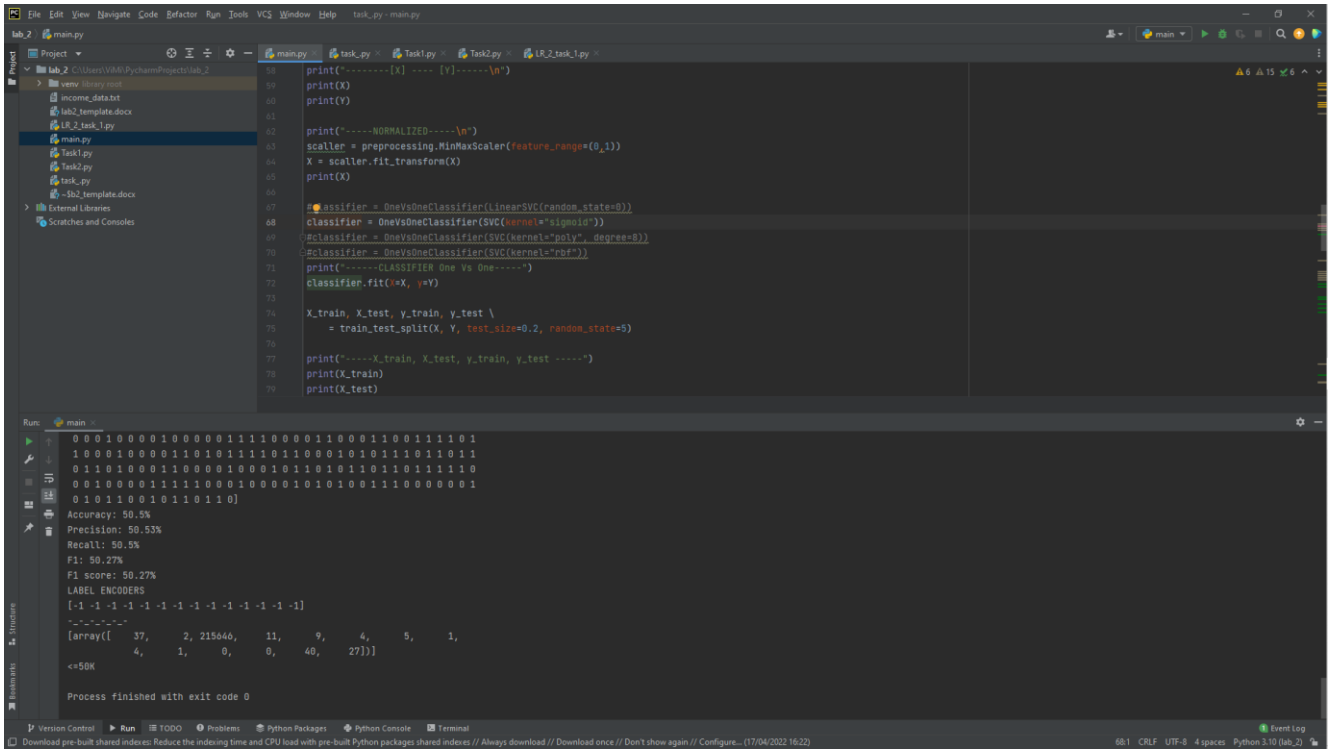

Рисунок 2. Сигмоїдне ядро

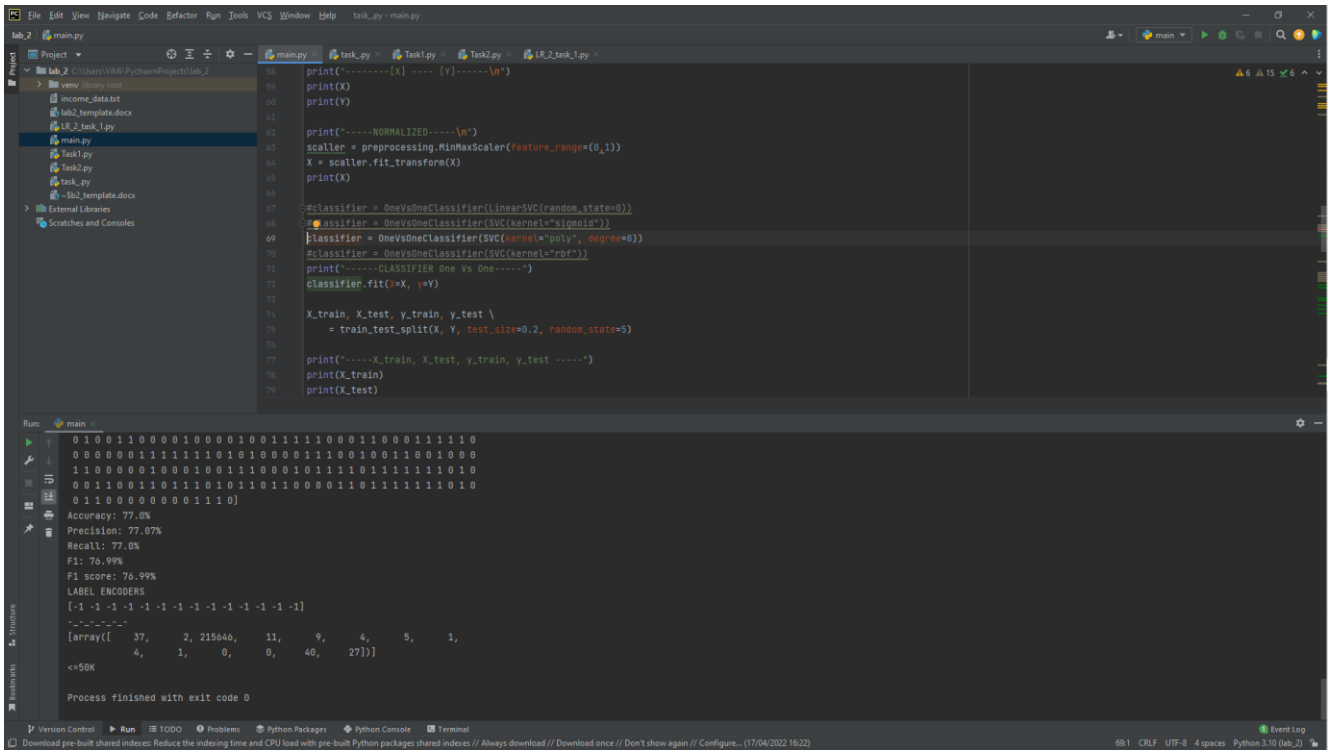

Рисунок 3. Poly ядро

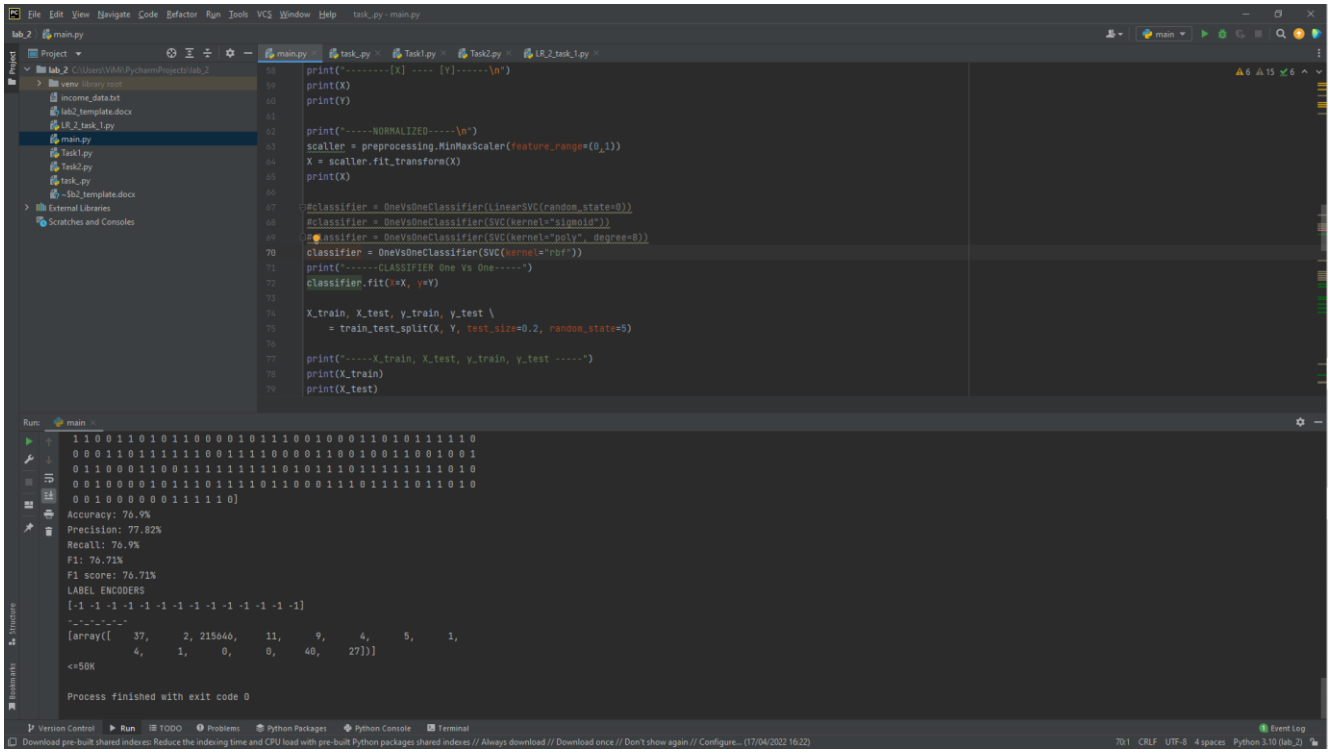

Рисунок 4. Rbf ядро

Як можна бачити rbf дало гарні результати дещо поступившись полі ядру, але набагато вигравши в швидкодії. Сигмоїдне ядро дало більш низькі результати, що дає нам основи вважати що rbf ядро, для нашого випадку, надає найкріші поєднання швидкодії та точності.

Завдання 3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Для виконання даного завдання використаємо набір даних що надається бібліотекою sklearn. Нижче наведено повний лістинг программи.

Лістинг програми:

```
from sklearn.datasets import load_iris
iris_dataset = load_iris()


print("Ключи Iris Dataset : \n{}".format(iris_dataset.keys()))
print(iris_dataset["DESCR"][:193] + "\n...")
print("Names answers: {}".format(iris_dataset["target_names"]))
print("Names for description: \n{}".format(iris_dataset["feature_names"]))
print("Type of data array: {}".format(type(iris_dataset["data"])))
print("Form of array data: {}".format(iris_dataset["data"].shape))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))

# Завантаження бібліотек
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


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))

# Стастичні 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)


#LOAD MODELS
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')))

#Quality models
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='accu-
racy')
    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))
```

При виконанні роботи оброробивши модель ми отримали наступну діаграмму розміху ірисів(рис.5)

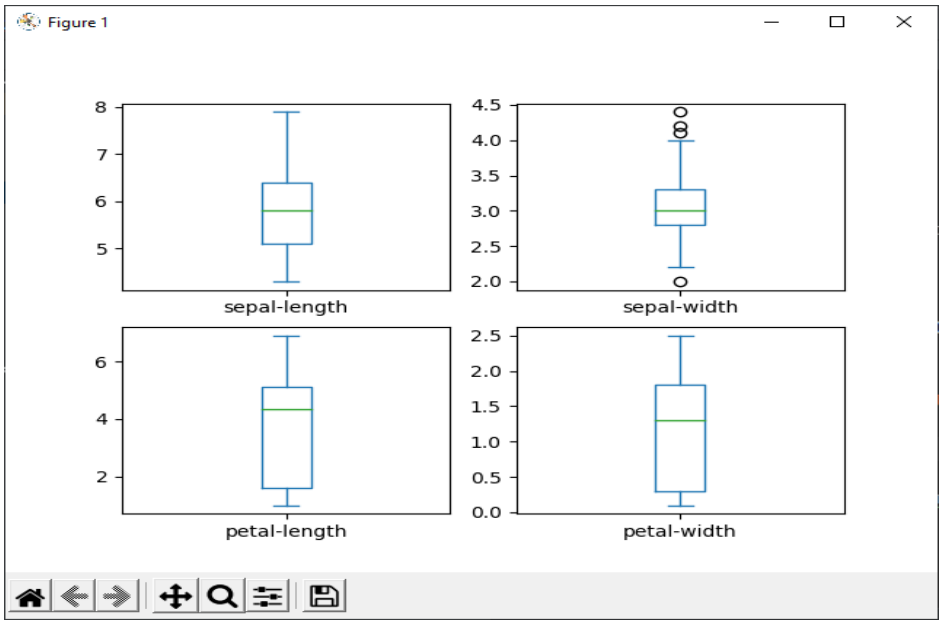

Рисунок 5. Діаграмма розмаху

Гістограмму розподілу(рис.6)

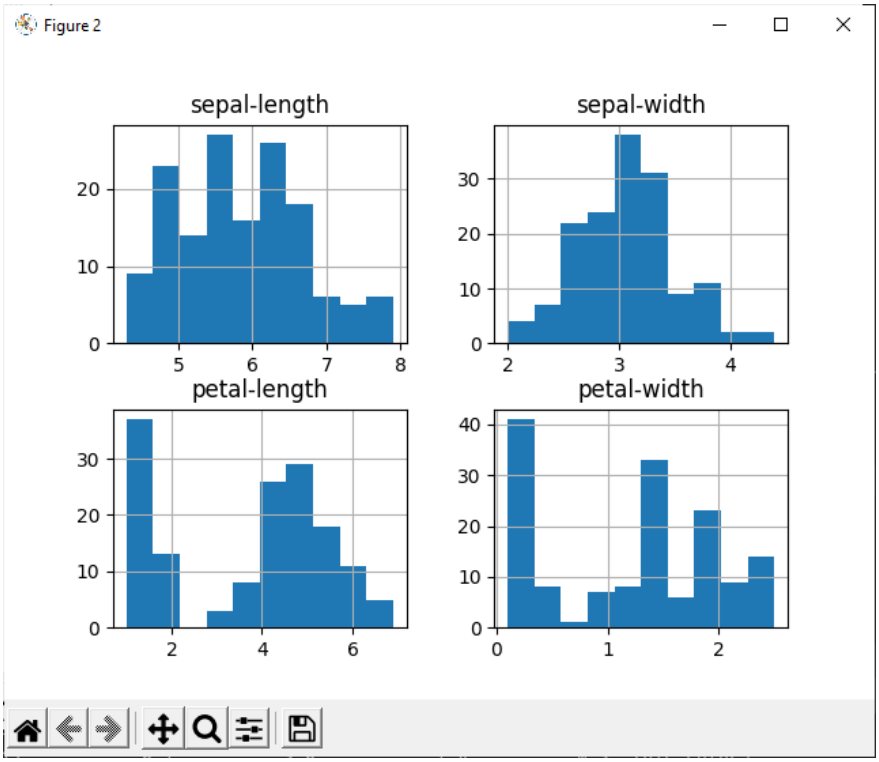

Рисунок 6. Гістограмма розподілу атрибутів датасет

Та матрицю діагрми розсіювання (рис.7)

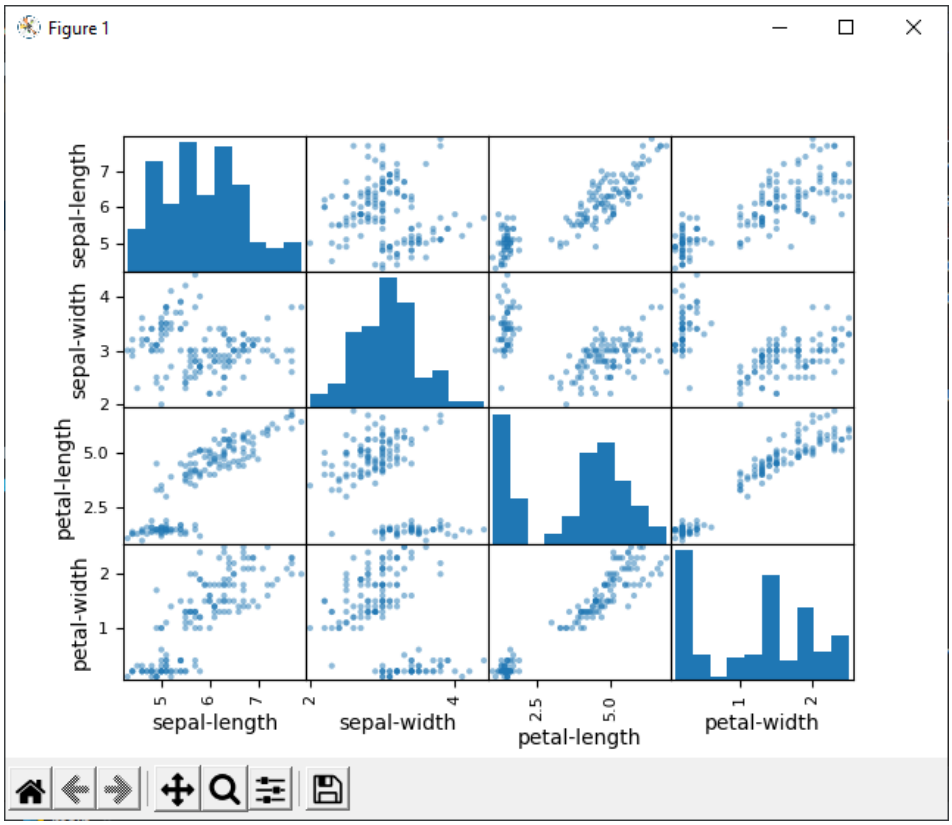

Рисунок 7. Матриця діаграми розсіювання

Сформувавши оцінки алгоритмів було отримано наступну діаграму для поточнох конфігурації

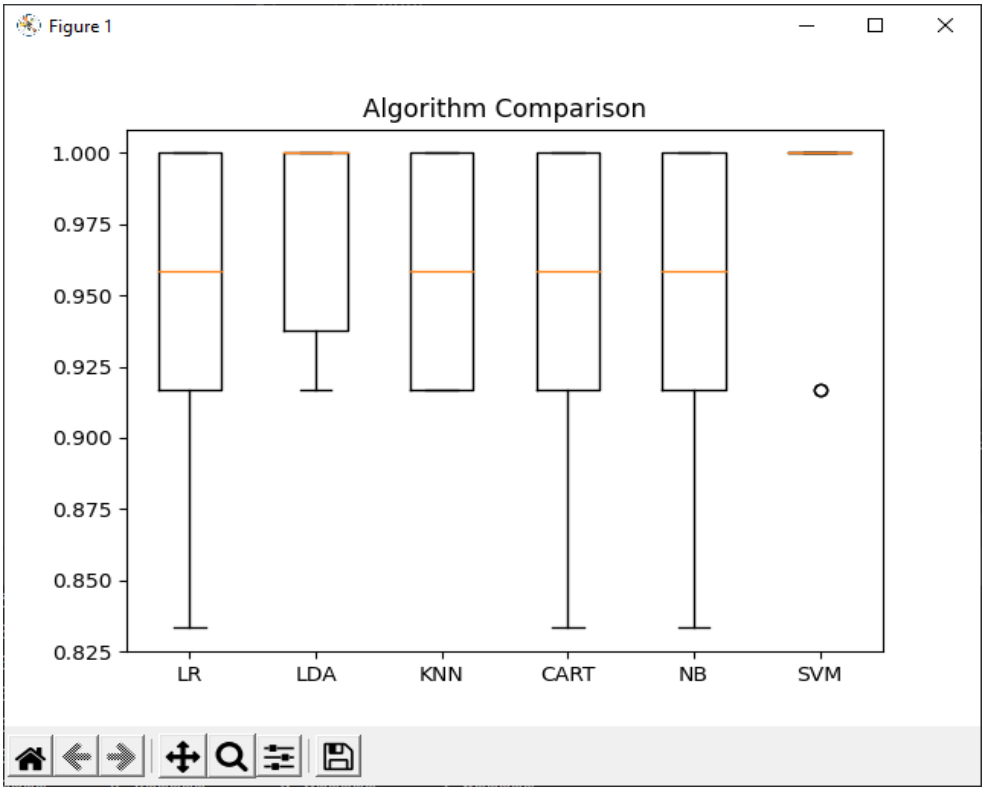

Рисунок 7. Порівняння алгоритмів

Як можна побачити з діаграми порівняння алгоритмів для конфігурації студента Найкраще себе проявила модель лінійного дискрімінінтного аналізу однак вона не досить стійка й під час тестування займала порівнянно більше часу на своє виконання. Досить непогані результати було отримано з алгоритму к сусідів, але найкраще поєднання передбачиваної точності й швидкої дає алгоритм дерева рішень.

Наступним кроком проведемо оцінки моделі та застосуємо модель для передбачення (код програми засновано на прикладі з методичних рекомендацій з дописом деяких частин)

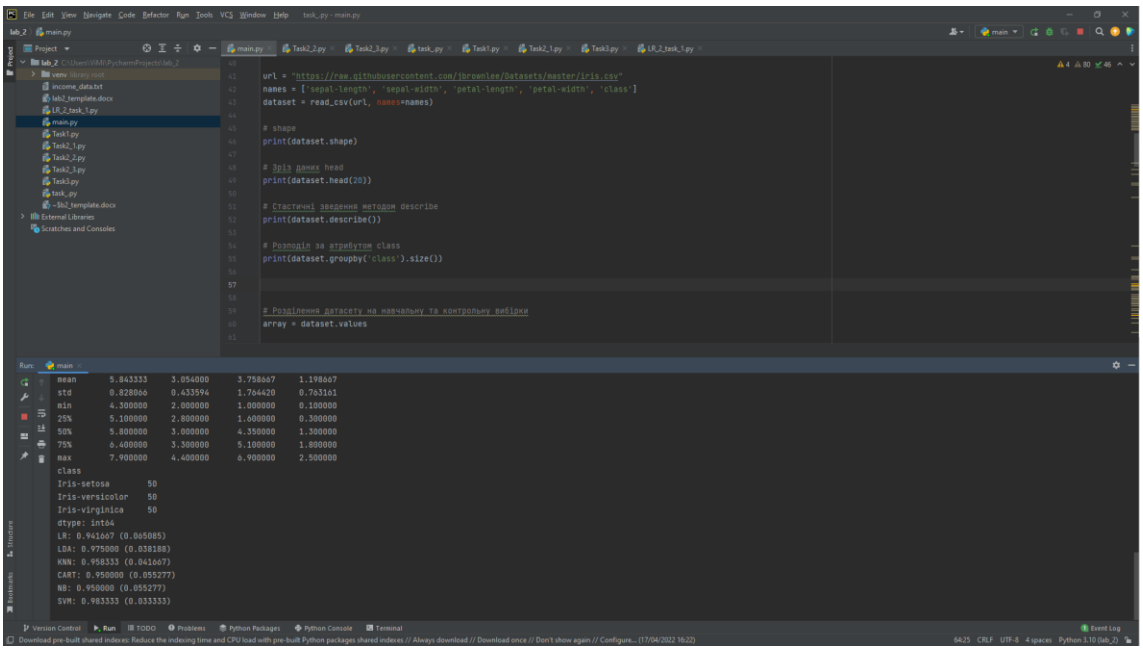

Рисунок 9. Обробка якостімоделі

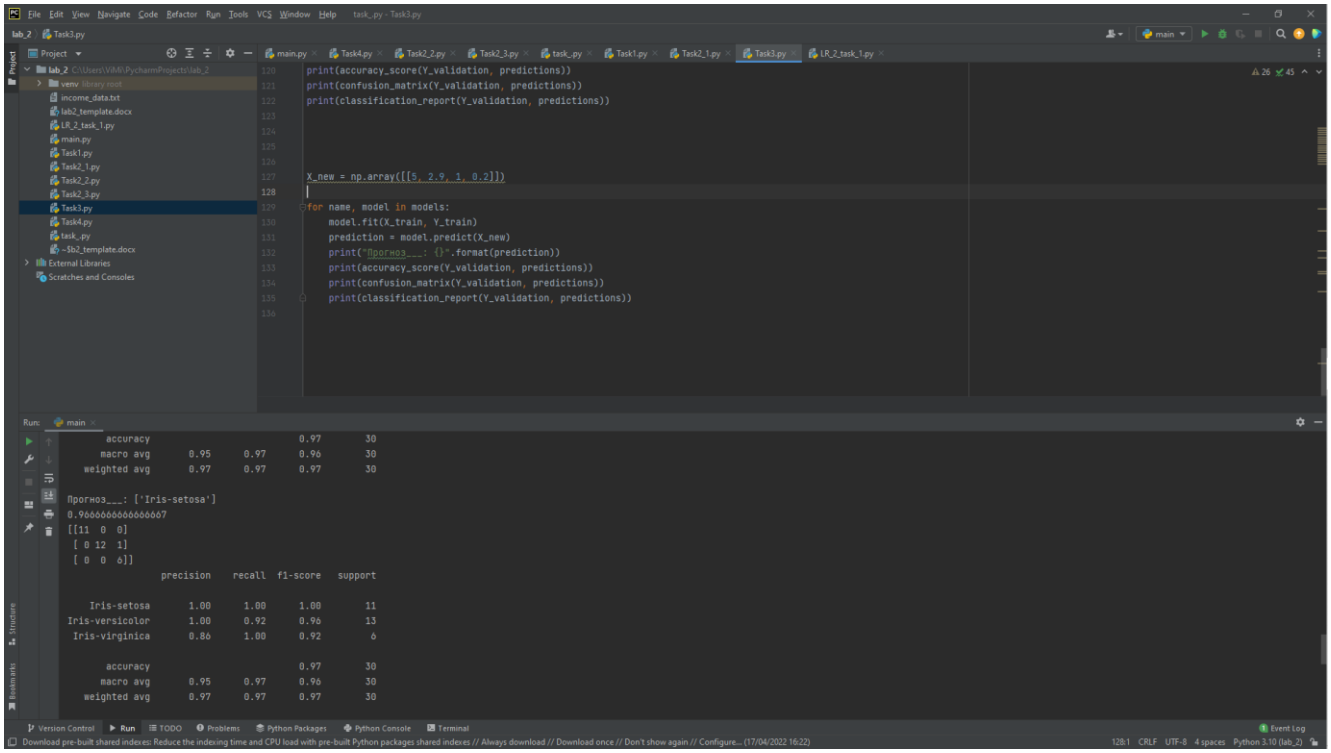

Рисунок 10. Використання моделі для передбачення

Квітка з кроку належить до класу setosa

Завдання 4: Порівняння якості класифікаторів для набору даних

Для виконання цього завдання адаптуємо з виокристанням багатьох моделей до використання на першому наборі даних.

Лістинг програми:

```
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


import numpy as np
import matplotlib.pyplot as plt
import sklearn.svm
from sklearn import preprocessing
from sklearn.svm import LinearSVC
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.metrics import accuracy_score
```

```
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score


#Input Data
input_file = "income_data.txt"

X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 500


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)


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')))

for name, model in models:
    classifier = OneVsOneClassifier(model)
    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)
    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='preci-
sion_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))

    print("LABEL ENCODERS")
    print(input_data_encoded)
    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 ]

    print("-_-_-_-_-_-_-")
    print(input_data_encoded)

    predicate_class = classifier.predict(input_data_encoded)
    print(label_encoder[-1].inverse_transform(predicate_class)[0])
```

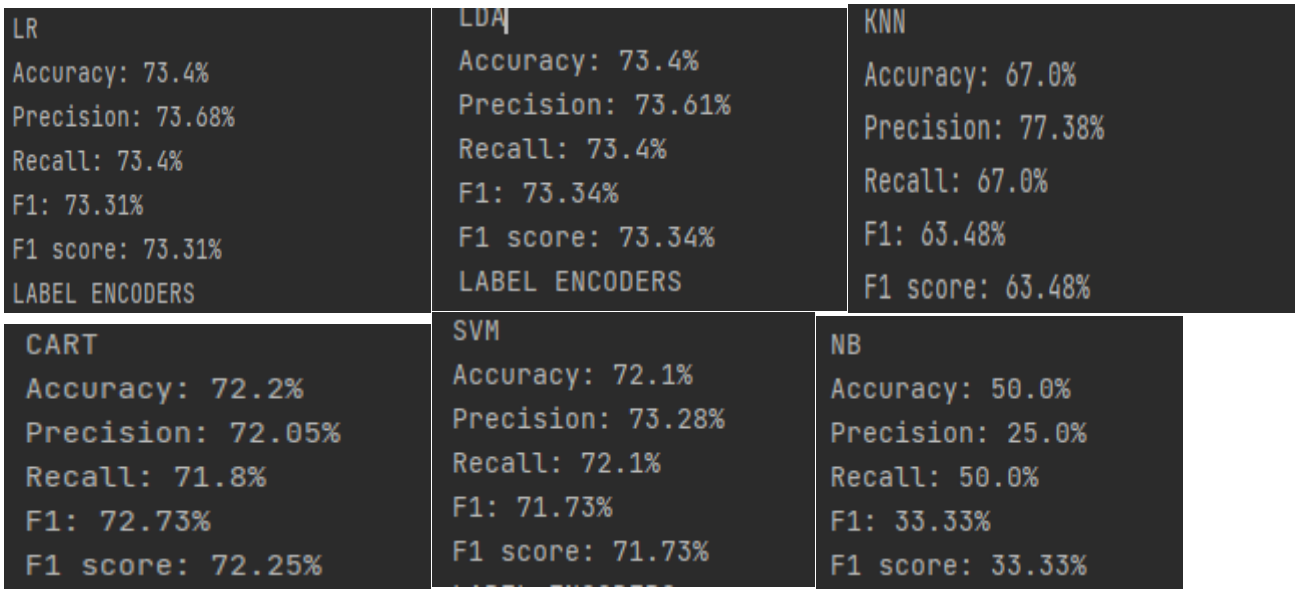

Рисунок 11. Отримані результати точності для різних класифікаторів

Завдання 5: Класифікація данних лінійним класифікатором Ridge

Наступним кроком проведемо класифікаю за допомогою лынійного класифікатора Ridge

Лістинг програми:

```
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

#LAST TASK
# ================================================
# Приклад класифікатора Ridge
# ================================================================================
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier

iris = load_iris()
X, y = iris.data, iris.target
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.3, ran-
dom_state = 0)
clf = RidgeClassifier(tol = 1e-2, solver = "sag")
clf.fit(Xtrain,ytrain)
ypred = clf.predict(Xtest)

from sklearn import metrics
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))
print('\t\tClassification Report:\n', metrics.classification_report(ypred,ytest))

from sklearn.metrics import confusion_matrix
from io import BytesIO

#neded for plot
import seaborn as sns;
sns.set()
import matplotlib.pyplot as plt

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")
```

Використаємо два параметри налаштування класифікатора точність рішення (тол) та спосіб рішення що використовується при розрахунках (в нашому випадку це sag (Алгоритм середнього стохастичного градієнту)

```
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       0.44      0.89      0.59         9
           2       0.91      0.50      0.65        20

    accuracy                           0.76        45
   macro avg       0.78      0.80      0.75        45
weighted avg       0.85      0.76      0.76        45
```

Рисунок 12. Отриманий результат

Як можна побачити з отриманного результату було використано показники якості точність, ф1, recall, коефіцієнт Коена Каппа (статистичне значення, що вимірую міжрегаональну згоду на категоріальні предмети, вважається більш надійним ніж розрахунок у відсотках бо враховує випадковість) та коефіцієнт кореляції Метьюза (використовується в машиному навчанні як міра якості бінарних мультикласних класифікацій, враховує істинні, хибні, позитивні та негативні результати, може бути використанний коли класи мають дуже різні розміри)
Отсанні кроком пояснимо що наведено на малюнку нижче (рис.13)

Це – Матриця невідповідності, згідно інформації отриманій з мережі інтернет це спеціальна матриця що використовується в галузі машинного навчання, й зокрема в задачі статистичної класифікації, ма́триця невідпові́дностей(англ. confusion matrix), також відома як матриця помилок (англ. error matrix) - це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму, зазвичай керованого навчання (у спонтаннім навчанні її зазвичай називають ма́трицею допасо́ваності, англ. matching matrix). 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 (наприклад, часто помилково маркуючи один як інший).

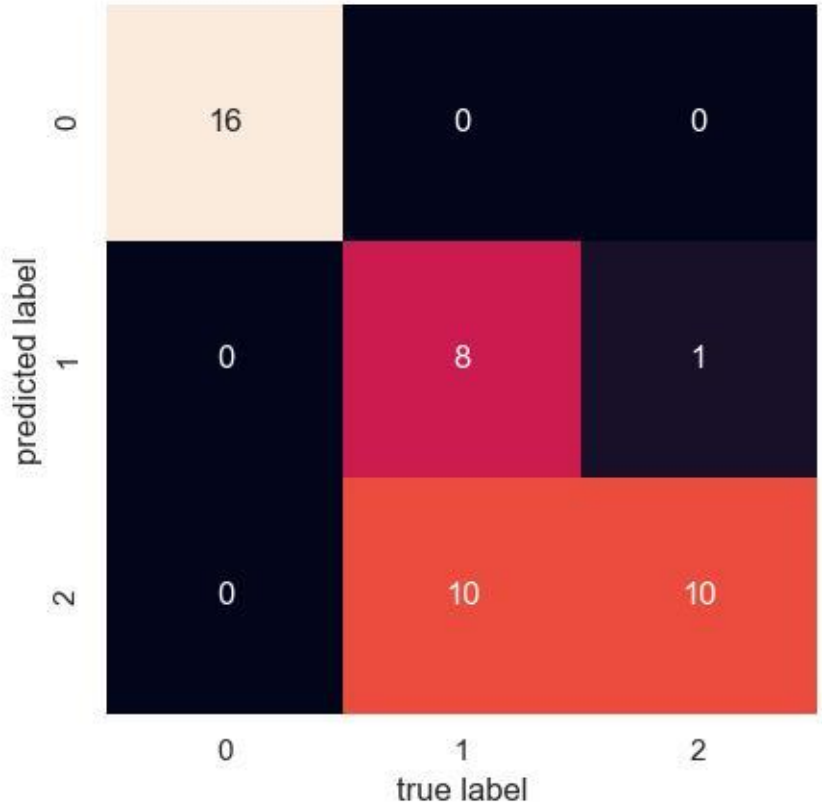

Рисунок 13. Матриця невідповідностей

Висновок: Розглянули особливості роботи з машинами векторів. Провели порівняльний аналіз різних методів класифікації даних.