

ЛАБОРАТОРНА РОБОТА № 2

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

Github: [link](#)

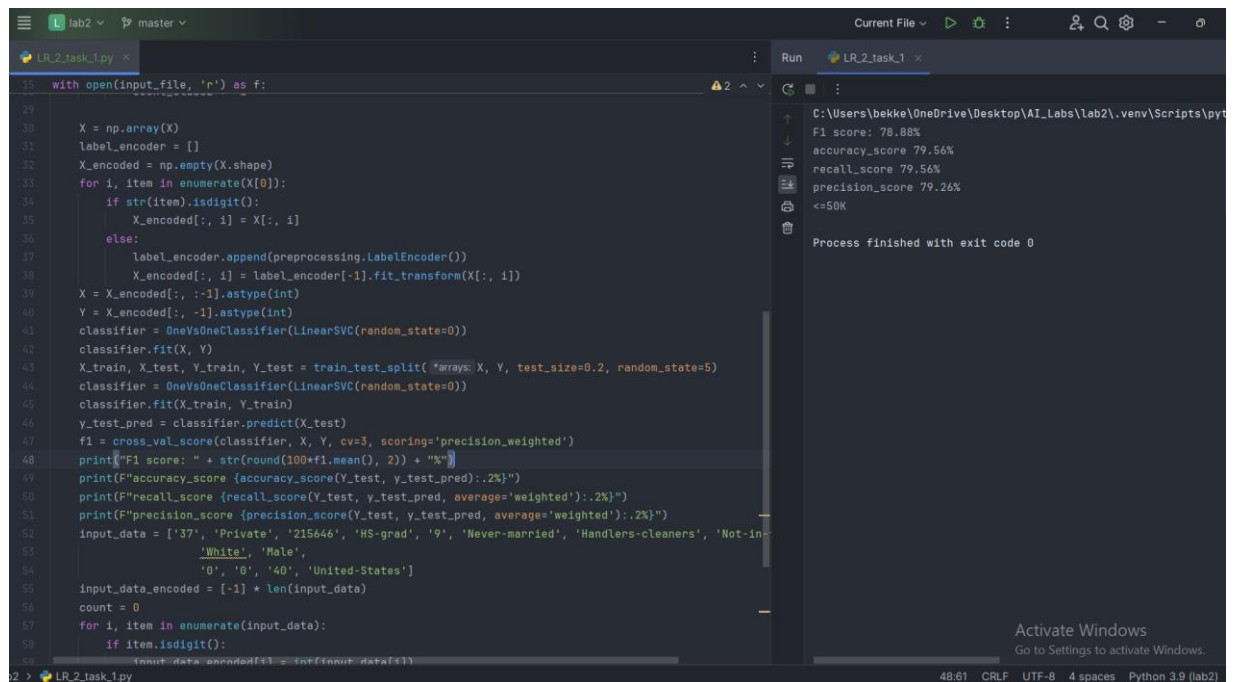
Завдання 1. Класифікація за допомогою машин опорних векторів (SVM)

Випишіть у звіт всі 14 ознак з набору даних – їх назви та що вони позначають та вид (числові чи категоріальні).

Column	Description	Example Value	Data Type
1	Age	39 , 50 , 38	Numeric
2	Work class	State-gov , Private	Categorical
3	Final weight (fnlwgt)	77516 , 83311	Numeric
4	Education	Bachelors , HS-grad	Categorical
5	Years of education	13 , 9 , 7	Numeric
6	Marital status	Never-married , Married-civ-spouse	Categorical
7	Occupation	Adm-clerical , Handlers-cleaners	Categorical
8	Relationship	Not-in-family , Husband , Wife	Categorical
9	Race	White , Black	Categorical
10	Sex	Male , Female	Categorical
11	Capital gain	2174 , 0	Numeric
12	Capital loss	0 , 0	Numeric
13	Hours per week	40 , 13 , 16	Numeric
14	Native country	United-States , Cuba , Jamaica	Categorical
15	Income category	<=50K , >50K	Categorical

Рис 1. Ознаки

Обчисліть значення інших показників якості класифікації (акуратність, повнота, точність) та разом з F1 занесіть їх у звіт.



The screenshot shows a Jupyter Notebook interface with a file named 'LR_2_task_1.py'. The code defines a function to load data from 'income_data.txt', preprocess it using a LabelEncoder, and train a LinearSVC classifier. It then evaluates the classifier using cross-validation and prints several performance metrics. The output on the right shows the following results:

Metric	Value
F1 score	78.88%
accuracy_score	79.56%
recall_score	79.56%
precision_score	79.26%

The process finished with exit code 0.

Рис 2. Розрахунки

Код програми занесіть у звіт.

```
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, cross_val_score
from sklearn.metrics import accuracy_score, recall_score, precision_score

input_file = 'income_data.txt'

X = []
Y = []

count_class1 = 0
count_class2 = 0
max_datapoints = 25_000

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 str(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(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, cv=3, scoring='precision_weighted')
print(print("F1 score: " + str(round(100*f1.mean(), 2)) + "%"))
print(F"accuracy_score {accuracy_score(Y_test, y_test_pred):.2%}")
print(F"recall_score {recall_score(Y_test, y_test_pred, average='weighted'):.2%}")
print(F"precision_score {precision_score(Y_test, y_test_pred, average='weighted'):.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]])[0])
        count += 1
input_data_encoded = np.array(input_data_encoded).reshape(1, 14)

```

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

Зробіть висновок до якого класу належить тестова точка

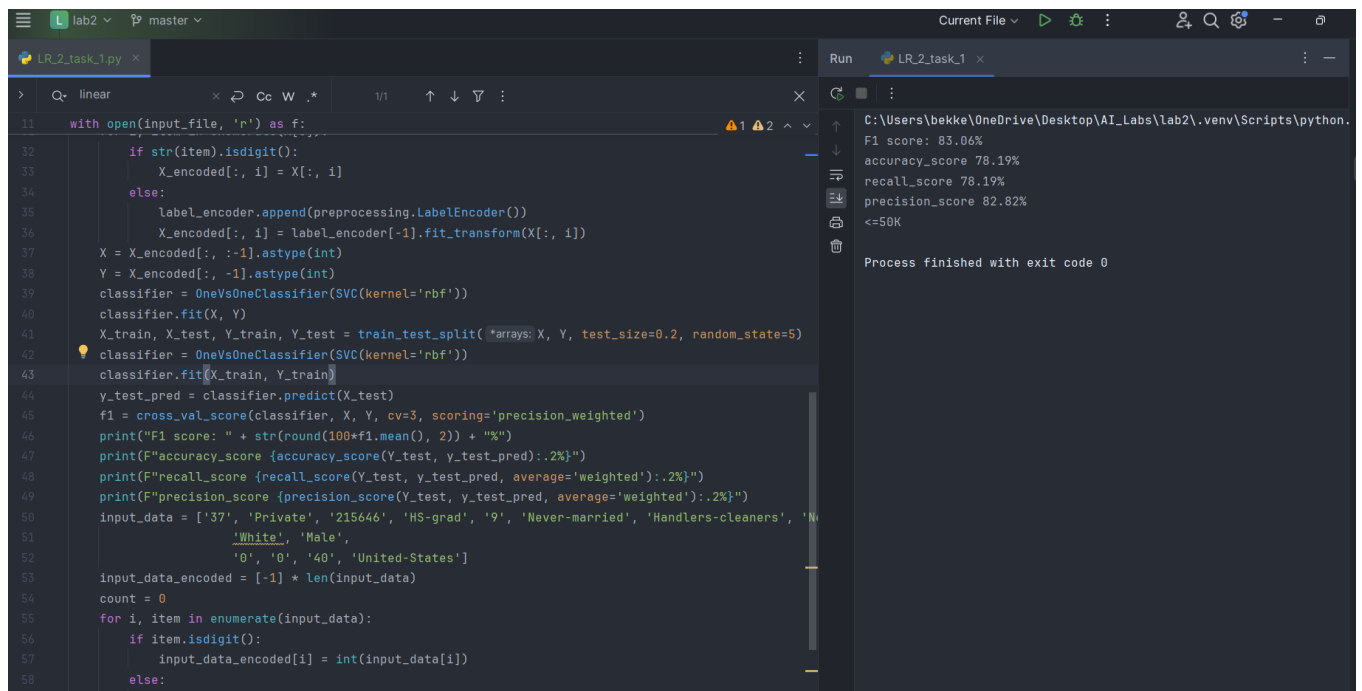
Тестова точка, відноситься до класу людей, у яких дохід менше або рівне 50K

Завдання 2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Поліноміальне ядро

Розрахунки проводяться довго, результатів досягнути не вдалося.

Гаусове ядро



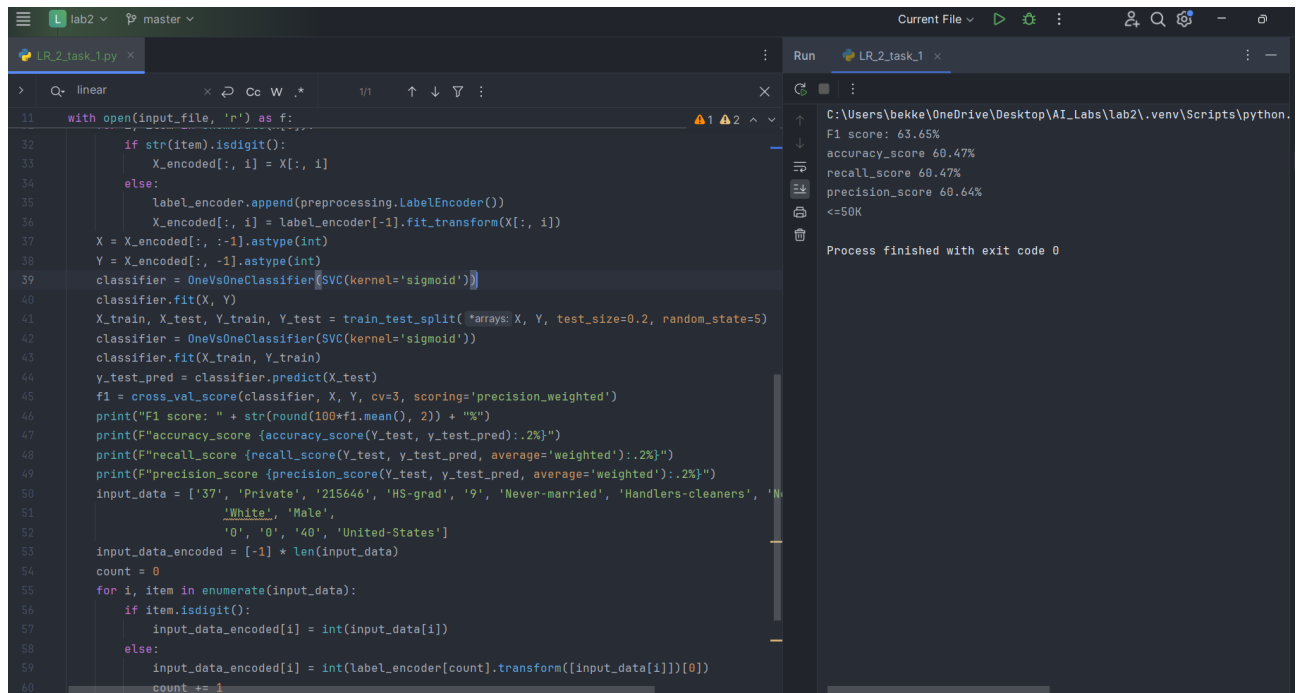
The screenshot displays a Jupyter Notebook interface with a file named 'LR_2_task_1.py'. The code implements an SVM classifier using a Gaussian kernel ('rbf'). It includes data preprocessing, model training, and evaluation metrics. The output console shows the following results:

```
F1 score: 83.06%
accuracy_score 78.19%
recall_score 78.19%
precision_score 82.82%
<=50K
```

The process finished with exit code 0.

Рис 3. Гаусове ядро

Сигмоїдальне ядро



```
11 with open(input_file, 'r') as f:
12     if str(item).isdigit():
13         X_encoded[:, 1] = X[:, 1]
14     else:
15         label_encoder.append(preprocessing.LabelEncoder())
16         X_encoded[:, 1] = label_encoder[-1].fit_transform(X[:, 1])
17 X = X_encoded[:, :-1].astype(int)
18 Y = X_encoded[:, -1].astype(int)
19 classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))
20 classifier.fit(X, Y)
21 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=5)
22 classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))
23 classifier.fit(X_train, Y_train)
24 y_test_pred = classifier.predict(X_test)
25 f1 = cross_val_score(classifier, X, Y, cv=3, scoring='precision_weighted')
26 print("F1 score: " + str(round(100*f1.mean(), 2)) + "%")
27 print(F"accuracy_score {accuracy_score(Y_test, y_test_pred):.2%}")
28 print(F"recall_score {recall_score(Y_test, y_test_pred, average='weighted'):.2%}")
29 print(F"precision_score {precision_score(Y_test, y_test_pred, average='weighted'):.2%}")
30 input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'N
31             'White', 'Male',
32             '0', '0', '40', 'United-States']
33 input_data_encoded = [-1] * len(input_data)
34 count = 0
35 for i, item in enumerate(input_data):
36     if item.isdigit():
37         input_data_encoded[i] = int(input_data[i])
38     else:
39         input_data_encoded[i] = int(label_encoder[count].transform([input_data[i]])[0])
40         count += 1
```

Run LR_2_task_1

C:\Users\bekke\OneDrive\Desktop\AI_Labs\lab2\.venv\Scripts\python.
F1 score: 63.65%
accuracy_score 60.47%
recall_score 60.47%
precision_score 60.64%
<=50K
Process finished with exit code 0

Рис 4. Сигмоїдальне ядро

Kernel	F1	Accuracy	Recall	Precision	Predict
Linear	78.88	79.56	79.56	79.26	<= 50
Poly	None	None	None	None	None
Rbf	83.06	78.19	78.19	82.82	<= 50
Sigmoid	63.65	60.47	60.47	60.64	<= 50

Висновок: відповідно до показників, найкраще виконує завдання Гаусовське та Лінійне ядра, але всі ці ядра мають однакові результати: <= 50

Рис 6. Структура даних та результати виконання

```
1 from sklearn.datasets import load_iris
2 iris_dataset = load_iris()
3 print(f"Name of iris dataset: {iris_dataset['name']}")
4 print(f"Full description: {iris_dataset['DESCR']}")
5 """
6 print(f"Назва даної: {iris_dataset['name']}")
7 print(f"Назва ознак: {iris_dataset['feature_names']}")
8 print(f"Тип масиву data: {iris_dataset['data'].shape}")
9 print(f"Формат масиву data: {iris_dataset['data'].dtype}")
10 print(f"Data count: {len(iris_dataset['data'])}")
11 print(f"First 5 samples: {iris_dataset['data'][:5]}")
12 print(f"Тип масиву target: {iris_dataset['target']}")
13 print(f"Бігловий: {iris_dataset['target_names']}")
14 """
```

Run L2_task_3_current

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class:
- Iris-Setosa
- Iris-Versicolour
- Iris-Virginica

:Summary Statistics:

	Min	Max	Mean	SD	Class Correlation
sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

:Missing Attribute Values: None
:Class Distribution: 33.3% for each of 3 classes.
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in the ICI

Рис 7. Ознайомлення зі структурою даних

Створіть новий файл Python та імпортуйте такі пакети. Тут записані всі необхідні бібліотеки

```
1 from pandas import read_csv
2 from pandas.plotting import scatter_matrix
3 from matplotlib import pyplot
4 from sklearn.model_selection import train_test_split
5 from sklearn.model_selection import cross_val_score
6 from sklearn.model_selection import StratifiedKFold
7 from sklearn.metrics import classification_report
8 from sklearn.metrics import confusion_matrix
9 from sklearn.metrics import accuracy_score
10 from sklearn.linear_model import LogisticRegression
11 from sklearn.tree import DecisionTreeClassifier
12 from sklearn.neighbors import KNeighborsClassifier
13 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
14 from sklearn.naive_bayes import GaussianNB
15 from sklearn.svm import SVC
16
17 url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
18 names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
19 dataset = read_csv(url, names=names)
20 print(dataset.shape)
21 print(dataset.head(5))
22 print(dataset.describe())
23 print(dataset.groupby('class').size())
24
25
```

Run L2_task_3_2_current_2py

C:\Users\bekke\OneDrive\Desktop\AI_Labs\lab2\.venv\Scripts\python.exe C:\Users\bekke\OneDrive\Desktop\AI_Labs\lab2\.venv\Scripts\python.exe C:\Users\bekke\OneDrive\Desktop\AI_Labs\lab2\.venv\Scripts\python.exe (150, 5)

	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

	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

class	count
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

dtype: int64

Process finished with exit code 0

Рис 8. Ознайомлення зі структурою даних за допомогою нових методів

Візуалізація даних

Одновимірні графіки

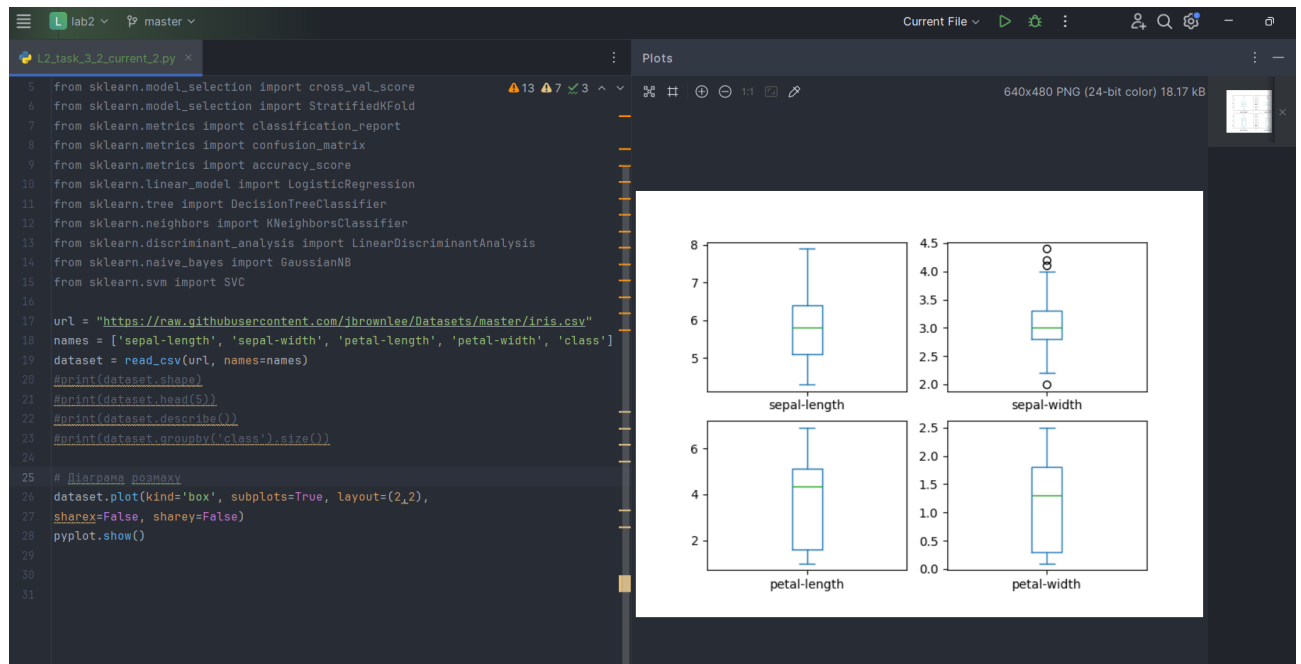


Рис 9. Box and whiskers diagram

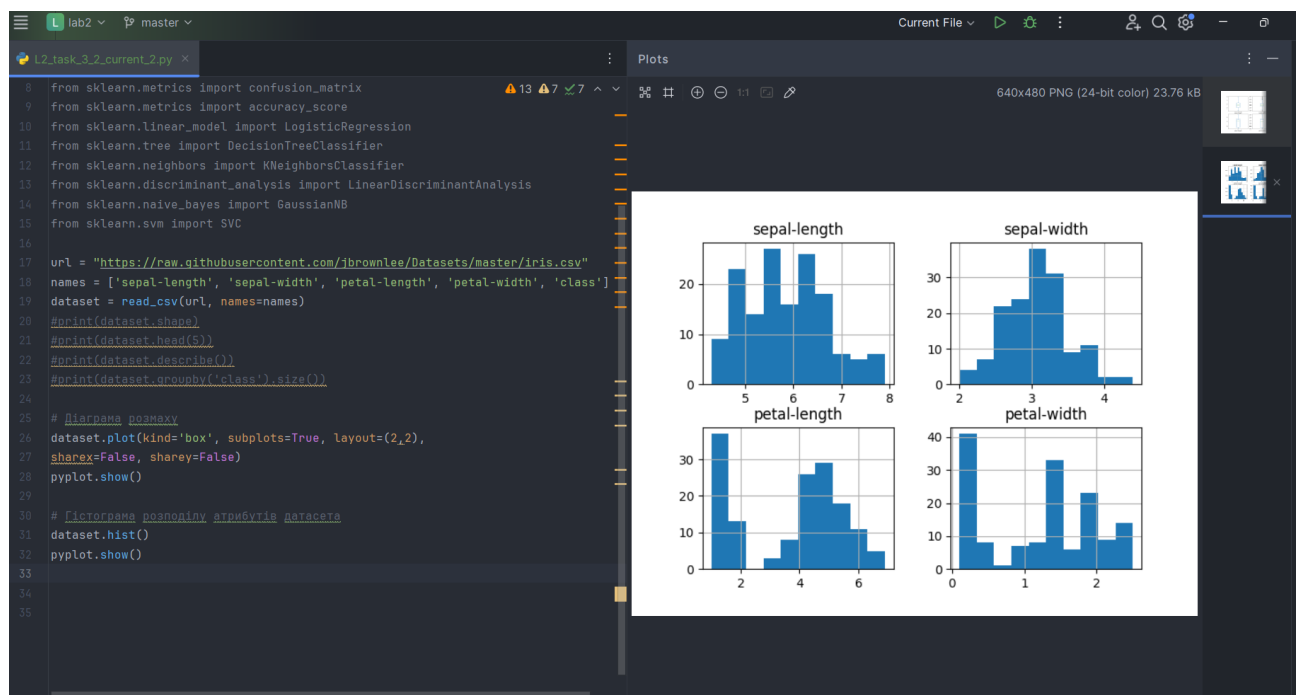


Рис 10. Гістограми

Багатовимірні графіки

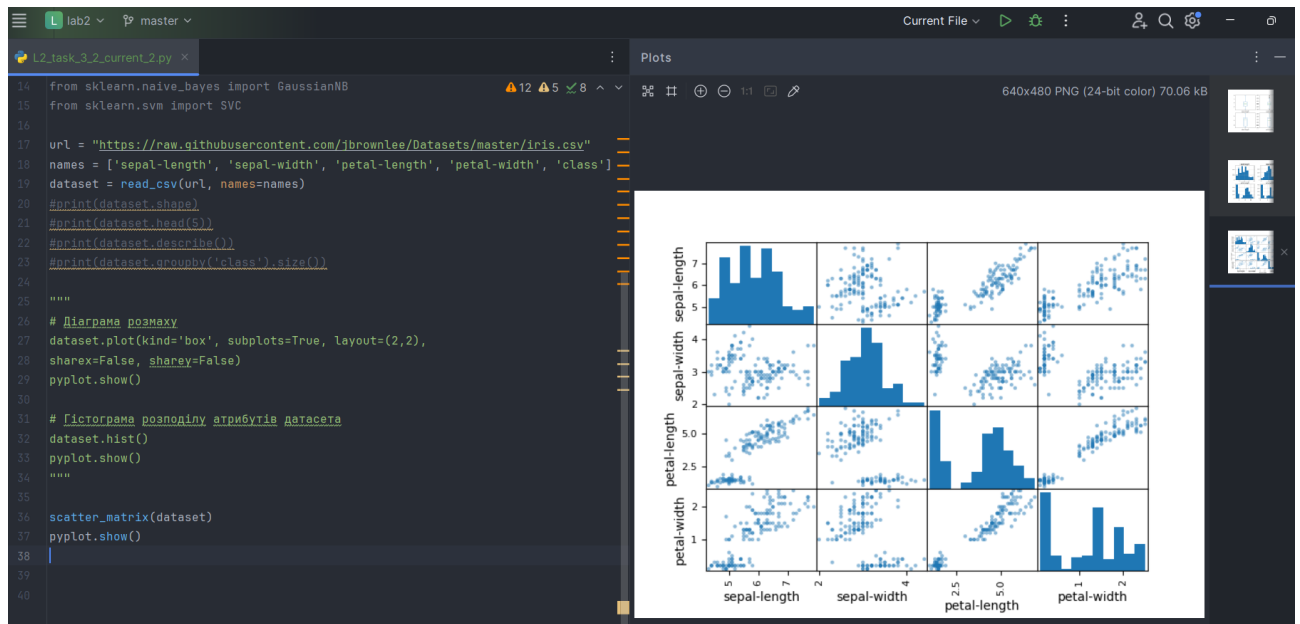


Рис 11. Scatterplot matrix

Створення навчального та тестового наборів

```
def split_data():
    array = dataset.values
    X = array[:, 0:4]
    y = array[:, 4]
    X_train, X_validation, Y_train, Y_validation = train_test_split(*arrays: X, y, test_size=0.20, random_state=1)
    print(X_train, X_validation, Y_train, Y_validation)
```

Рис 12. Створення навчального та тестового наборів

Класифікація (побудова моделі)

Отримані графіки та результати занесіть у звіт. Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте найкращим

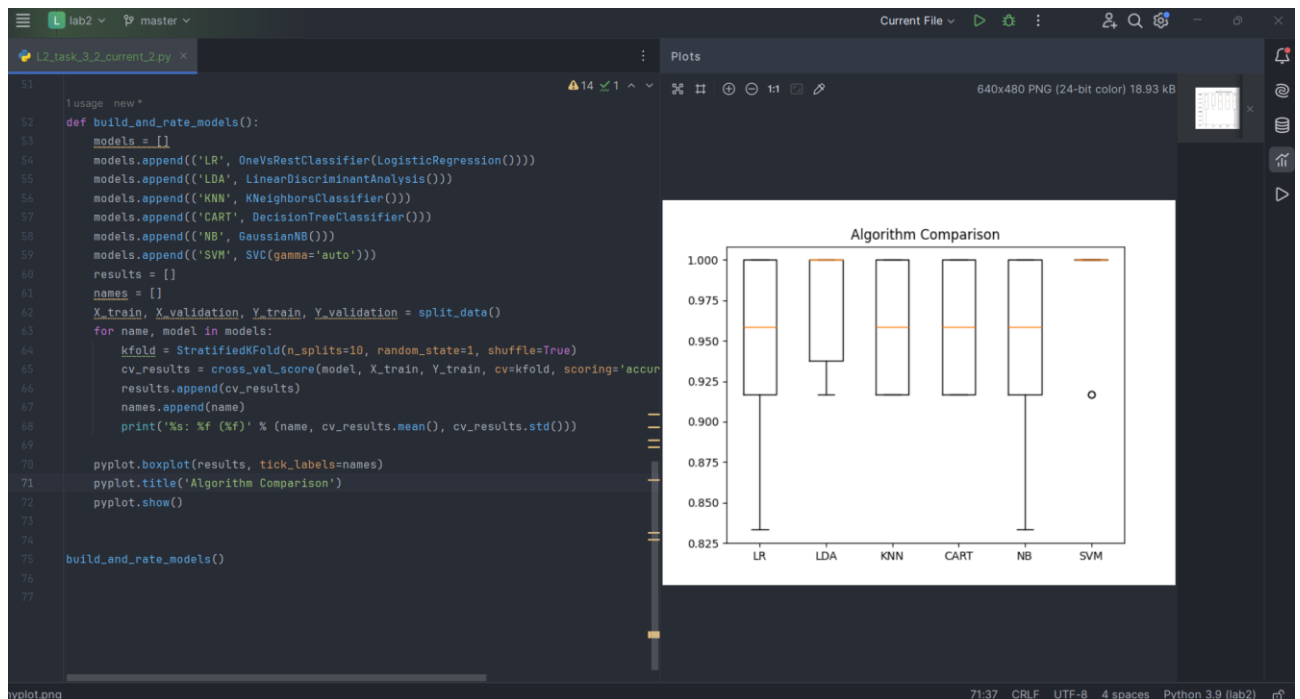


Рис 13. Box and whiskers diagram

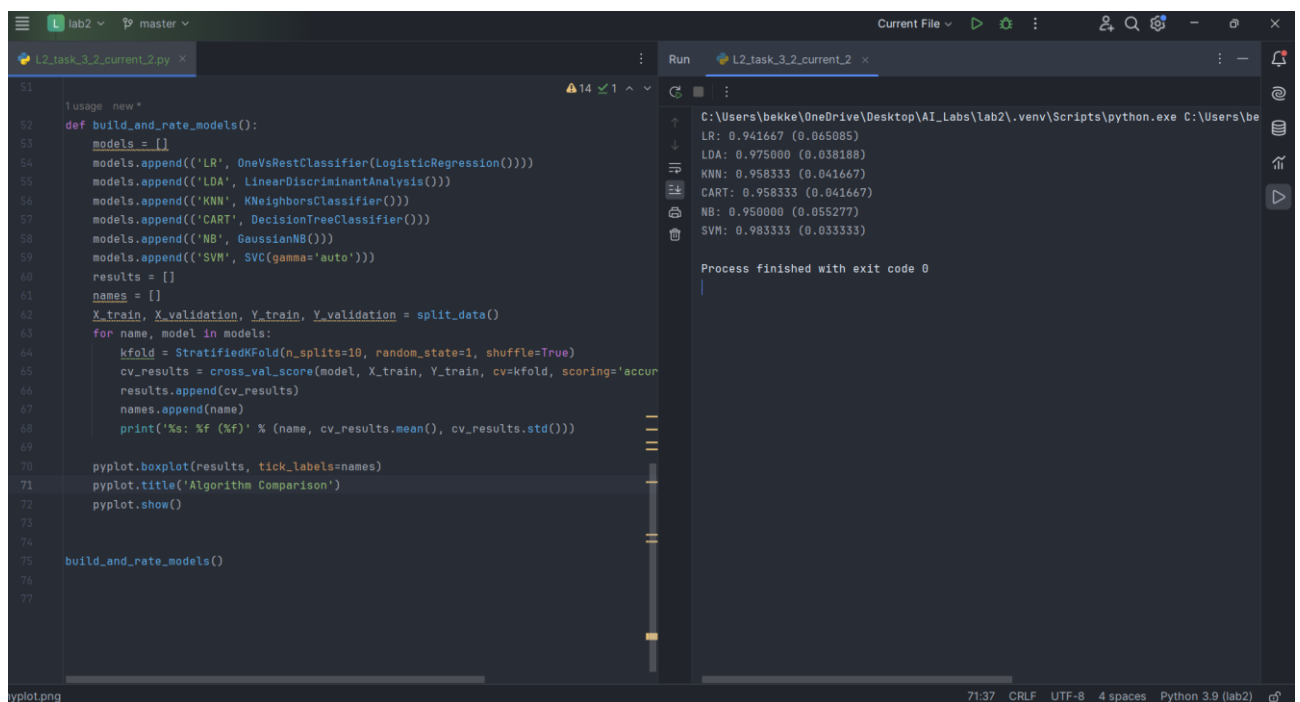


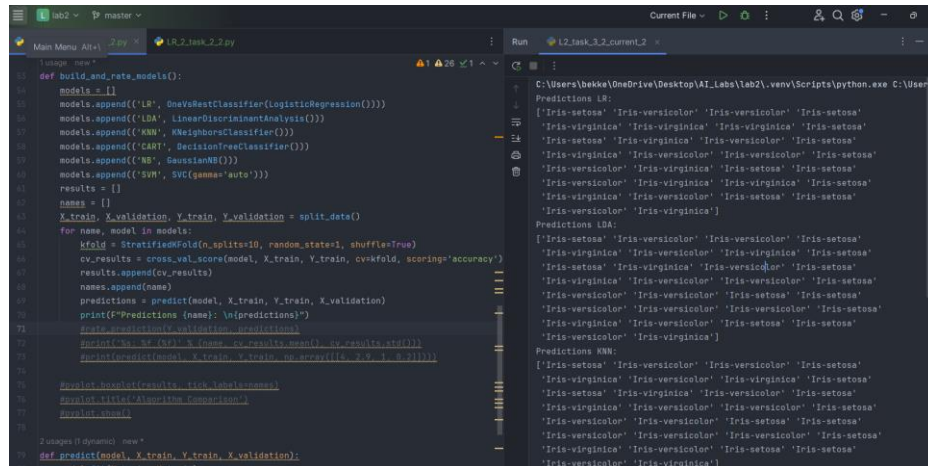
Рис 14. Розрахунки

Висновок: відповідно до розрахунків, найкращий метод – SVM, тому що він має найбільшу точність та найменше відхилення, але тут не враховані інші показники і значення інших моделей близькі за значеннями, тому важко обрати точно.

Відповідно до графіків, найкращим методом є також SVM, оскільки його boxplot виглядає як одна лінія, що означає, що метод є стабільним та забезпечує високу точність. Проте, є одне значення – викид, що свідчить про потенційні аномалії в деяких випадках.

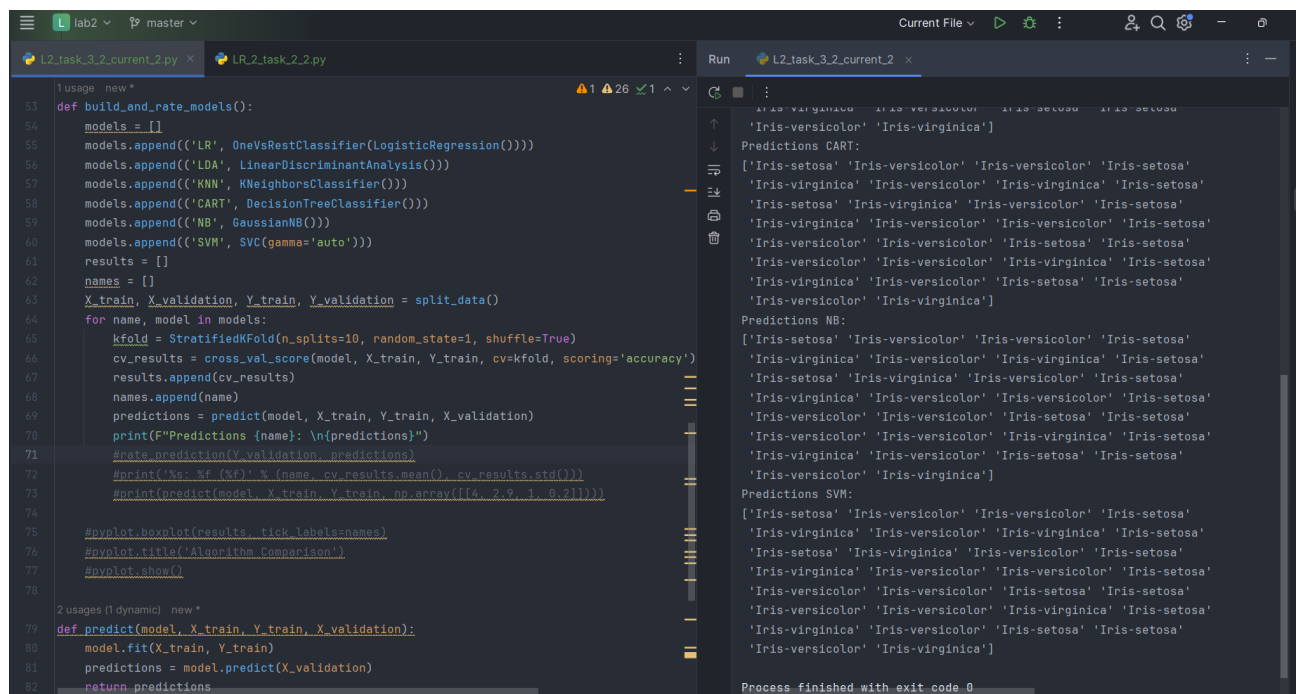
Крок 5. Оптимізація параметрів моделі

Крок 6. Отримання прогнозу (передбачення на тренувальному наборі)



```
def build_and_rate_models():
    models = []
    models.append(('LR', OneVsRestClassifier(LogisticRegression())))
    models.append(('LDA', LinearDiscriminantAnalysis()))
    models.append(('KNN', KNeighborsClassifier()))
    models.append(('CART', DecisionTreeClassifier()))
    models.append(('NB', GaussianNB()))
    models.append(('SVM', SVC(gamma='auto')))
    results = []
    names = []
    X_train, X_validation, Y_train, Y_validation = split_data()
    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)
        predictions = predict(model, X_train, Y_train, X_validation)
        print(f"Predictions {name}: \n{predictions}")
        #rate prediction(Y_validation, predictions)
        #print('%s %f (%f)' % (name, cv_results.mean(), cv_results.std()))
        #print(predict(model, X_train, Y_train, np.array([[4, 2.9, 1, 0.2]])))
        #pyplot.boxplot(results, tick_labels=names)
        #pyplot.title('Algorithm Comparison')
        #pyplot.show()
    #def predict(model, X_train, Y_train, X_validation):
    #    model.fit(X_train, Y_train)
    #    predictions = model.predict(X_validation)
    #    return predictions
```

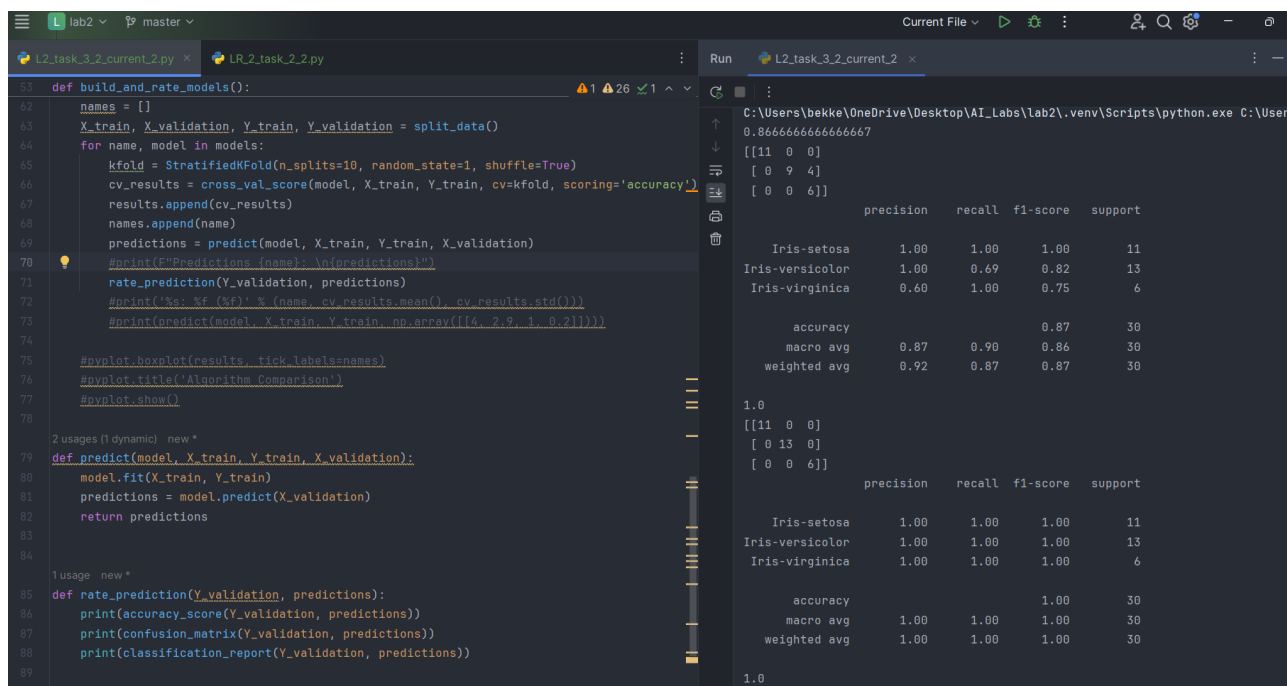
Рис 15. Отримання прогнозу



```
def build_and_rate_models():
    models = []
    models.append(('LR', OneVsRestClassifier(LogisticRegression())))
    models.append(('LDA', LinearDiscriminantAnalysis()))
    models.append(('KNN', KNeighborsClassifier()))
    models.append(('CART', DecisionTreeClassifier()))
    models.append(('NB', GaussianNB()))
    models.append(('SVM', SVC(gamma='auto')))
    results = []
    names = []
    X_train, X_validation, Y_train, Y_validation = split_data()
    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)
        predictions = predict(model, X_train, Y_train, X_validation)
        print(f"Predictions {name}: \n{predictions}")
        #rate prediction(Y_validation, predictions)
        #print('%s %f (%f)' % (name, cv_results.mean(), cv_results.std()))
        #print(predict(model, X_train, Y_train, np.array([[4, 2.9, 1, 0.2]])))
        #pyplot.boxplot(results, tick_labels=names)
        #pyplot.title('Algorithm Comparison')
        #pyplot.show()
    #def predict(model, X_train, Y_train, X_validation):
    #    model.fit(X_train, Y_train)
    #    predictions = model.predict(X_validation)
    #    return predictions
```

Рис 16. Отримання прогнозу

Крок 7. Оцінка якості моделі



```
53 def build_and_rate_models():
54     names = []
55     X_train, X_validation, Y_train, Y_validation = split_data()
56     for name, model in models:
57         kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
58         cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
59         results.append(cv_results)
60         names.append(name)
61         predictions = predict(model, X_train, Y_train, X_validation)
62         #print(F"Predictions {name}: \n{predictions}")
63         rate_prediction(Y_validation, predictions)
64         #print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
65         #print(predict(model, X_train, Y_train, np.array([[4, 2.9, 1, 0.2]])))
66
67         #pyplot.boxplot(results, tick_labels=names)
68         #pyplot.title('Algorithm Comparison')
69         #pyplot.show()
70
71 2 usages (1 dynamic) new *
72 def predict(model, X_train, Y_train, X_validation):
73     model.fit(X_train, Y_train)
74     predictions = model.predict(X_validation)
75     return predictions
76
77 1 usage new *
78 def rate_prediction(Y_validation, predictions):
79     print(accuracy_score(Y_validation, predictions))
80     print(confusion_matrix(Y_validation, predictions))
81     print(classification_report(Y_validation, predictions))
82
83 build_and_rate_models()
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	0.69	0.82	13
Iris-virginica	0.60	1.00	0.75	6
accuracy			0.87	30
macro avg	0.87	0.90	0.86	30
weighted avg	0.92	0.87	0.87	30

```
1.0
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]

precision    recall  f1-score   support

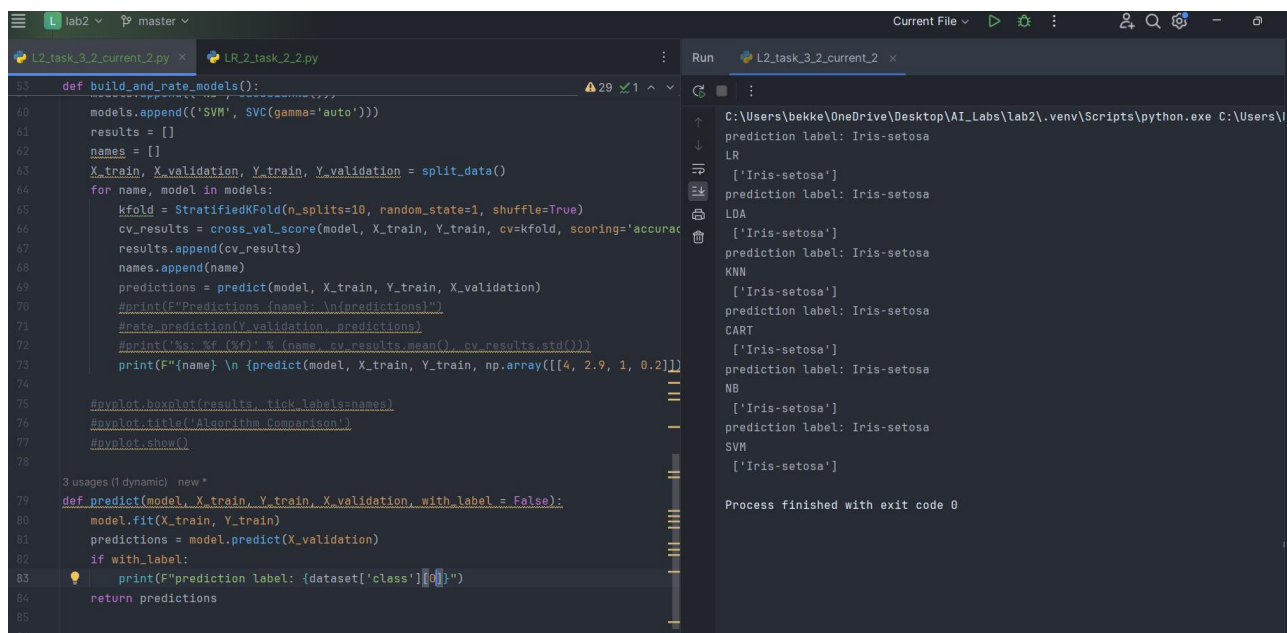
Iris-setosa      1.00      1.00      1.00        11
Iris-versicolor  1.00      0.69      0.82        13
Iris-virginica   0.60      1.00      0.75         6

 accuracy      0.87      0.90      0.87        30
 macro avg     0.87      0.90      0.86        30
 weighted avg   0.92      0.87      0.87        30
```

Рис 17. Оцінка якості моделей

Крок 8. Отримання прогнозу (застосування моделі для передбачення)

`print("Спрогнозованная метка: {}".format(iris_dataset['target_names'][prediction]))` – не зрозуміло, який результат очікується, тому на зображенні вивів назву класу.



```
53 def build_and_rate_models():
54     models.append(('SVM', SVC(gamma='auto')))
55     results = []
56     names = []
57     X_train, X_validation, Y_train, Y_validation = split_data()
58     for name, model in models:
59         kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
60         cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
61         results.append(cv_results)
62         names.append(name)
63         predictions = predict(model, X_train, Y_train, X_validation)
64         #print(F"Predictions {name}: \n{predictions}")
65         #rate_prediction(Y_validation, predictions)
66         #print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
67         #print(F"{name} \n {predict(model, X_train, Y_train, np.array([[4, 2.9, 1, 0.2]])}")
68
69         #pyplot.boxplot(results, tick_labels=names)
70         #pyplot.title('Algorithm Comparison')
71         #pyplot.show()
72
73 3 usages (1 dynamic) new *
74 def predict(model, X_train, Y_train, X_validation, with_label = False):
75     model.fit(X_train, Y_train)
76     predictions = model.predict(X_validation)
77     if with_label:
78         print(F"prediction label: {dataset['class'][0]}")
79     return predictions
80
81 build_and_rate_models()
82 predict(model, X_train, Y_train, X_validation, with_label = True)
```

```
prediction label: Iris-setosa
LR
['Iris-setosa']
prediction label: Iris-setosa
LDA
['Iris-setosa']
prediction label: Iris-setosa
KNN
['Iris-setosa']
prediction label: Iris-setosa
CART
['Iris-setosa']
prediction label: Iris-setosa
NB
['Iris-setosa']
prediction label: Iris-setosa
SVM
['Iris-setosa']

Process finished with exit code 0
```

Рис 18. Прогнози моделей

Висновок: Квітка за кроку 8 належить до класу 'Iris-setosa'. Якості класифікацій наведені в кроці 7.

4. Порівняння якості класифікаторів для набору даних завдання 2.1

Коди та результати занесіть у звіт:

```
from pandas import read_csv
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
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
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn import preprocessing

names = ['age', 'work-class', 'final-weight',
         'education', 'years-of-education', 'marital-status',
         'occupation', 'relationship', 'race',
         'sex', 'capital-gain', 'capital-loss',
         'hours-per-week', 'native-country', 'income'
        ]

dataset = read_csv('income_data.txt', names=names)
models = [('LR', OneVsRestClassifier(LogisticRegression())),
          ('LDA', LinearDiscriminantAnalysis()),
          ('KNN', KNeighborsClassifier()),
          ('CART', DecisionTreeClassifier()),
          ('NB', GaussianNB()),
          ('SVM', SVC(gamma='auto'))
         ]

ignore = ['LDA', 'KNN', 'CART', 'NB', 'SVM']
array = dataset.values

def encode():
    X_encoded = np.empty(array.shape)
    label_encoders = []
```

```

for i in range(array.shape[1] - 1):
    column = array[:, i]
    if str(column).isdigit():
        X_encoded[:, i] = column
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(column)
        label_encoders.append(le)

return (X_encoded[:, :-1].astype(int), array[:, -1])

def scale(X, y):
    X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20, random_state=1)
    scaler = preprocessing.StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_validation = scaler.transform(X_validation)
    return (X_train, X_validation, Y_train, Y_validation)

X, y = encode()
X_train, X_validation, Y_train, Y_validation = scale(X, y)
reports = []
for name, model in models:
    model.fit(X_train, Y_train)
    predictions = model.predict(X_validation)
    report = classification_report(Y_validation, predictions)
    print(report)

```

```
LR
precision    recall  f1-score   support

<=50K        0.85        0.94        0.89        5026
>50K         0.69        0.45        0.54        1487

accuracy          0.83        6513
macro avg         0.77        0.69        0.72        6513
weighted avg      0.82        0.83        0.81        6513

LDA
precision    recall  f1-score   support

<=50K        0.85        0.95        0.89        5026
>50K         0.70        0.42        0.52        1487

accuracy          0.83        6513
macro avg         0.77        0.68        0.71        6513
weighted avg      0.81        0.83        0.81        6513

KNN
precision    recall  f1-score   support

<=50K        0.89        0.89        0.89        5026
>50K         0.64        0.64        0.64        1487

accuracy          0.84        6513
macro avg         0.77        0.77        0.77        6513
weighted avg      0.84        0.84        0.84        6513

CART
precision    recall  f1-score   support

<=50K        0.89        0.87        0.88        5026
>50K         0.59        0.63        0.61        1487

accuracy          0.81        6513
macro avg         0.74        0.75        0.74        6513
weighted avg      0.82        0.81        0.82        6513
```

Рис 19. Оцінки якості моделей

NB					
	precision	recall	f1-score	support	
<=50K	0.86	0.93	0.89	5026	
>50K	0.66	0.47	0.55	1487	
accuracy			0.82	6513	
macro avg	0.76	0.70	0.72	6513	
weighted avg	0.81	0.82	0.81	6513	
SVM					
	precision	recall	f1-score	support	
<=50K	0.88	0.94	0.91	5026	
>50K	0.73	0.55	0.63	1487	
accuracy			0.85	6513	
macro avg	0.80	0.74	0.77	6513	
weighted avg	0.84	0.85	0.84	6513	

Рис 20. Оцінки якості моделей

Висновок: Мені важко визначити яку задачу ми вирішували, тому й важко обрати найкращу модель. Якщо задачею було знайти класифікатор, який краще визначає людей з ≤ 50 , то це буде один класифікатор, якщо > 50 , то другий, якщо потрібно в середньому визначати, то третій. Можливо, існують інші зовнішні критерії, які також потрібно врахувати.

Загалом, також важко порівнювати репорти, тому що їх багато, тому потрібно писати додаткові функції, які оцінять найкращу модель, але цього не зазначено в завданні.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають

Tol - означає допуск на збіжність. Коли різниця в коефіцієнтах (вагах) між ітераціями стає меншою за це значення, розв'язувач завершує цикл. Це вказує на те, наскільки точною є процедура оптимізації.

Solver - розв'язувач для використання в обчислювальних процедурах. Значення sag використовує метод стохастичного середнього градієнта, а «saga» - його неупереджену та гнучкішу версію під назвою SAGA. Обидва методи використовують ітераційну процедуру і часто є швидшими за інші розв'язувачі, коли $n_{\text{вибірок}}$ та $n_{\text{особливостей}}$ є великими.

Опишіть які показники якості використовуються та їх отримані результати. Вставте у звіт та поясніть зображення Confusion.jpg

Показник	Значення
Accuracy	0.7556
Precision	0.8333
Recall	0.7556
F1 Score	0.7503
Cohen Kappa Score	0.6431
Matthews Corrcoef	0.6831

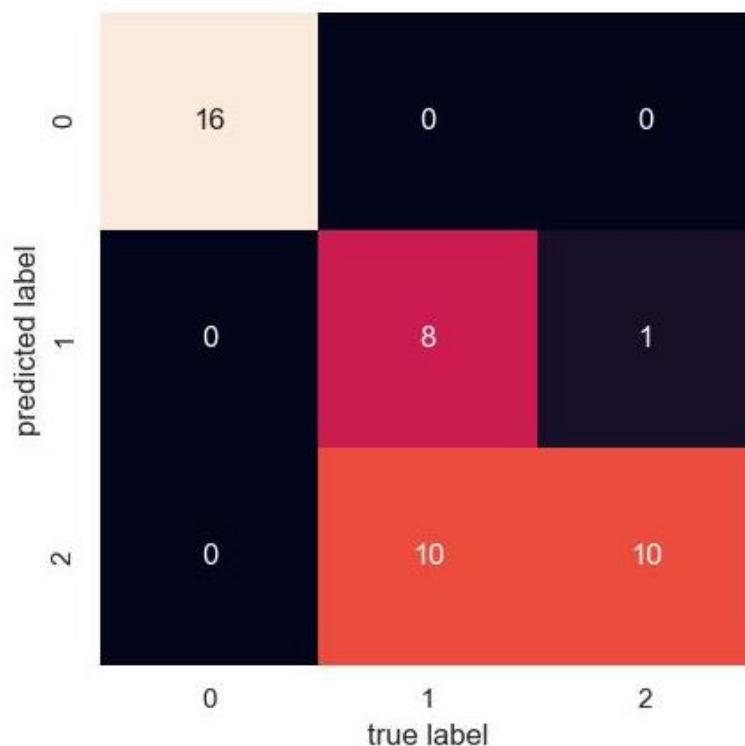


Рис 21. Confusion matrix

На даному зображенні по осі X відображенні правильні відповіді. (правильні/вірні класи), а по осі Y передбачені класифікатором чи моделлю класи (відповіді), тому:

Для «true label» 0, всі 16 екземплярів даних були правильно класифіковані як 0

Для «true label» 1, 10 екземплярів було неправильно класифіковано як 2 та 8 правильно класифікованих як 1

Для «true label» 2, 10 екземплярів даних було правильно класифіковано як 2 та 1 неправильно класифікований як 1

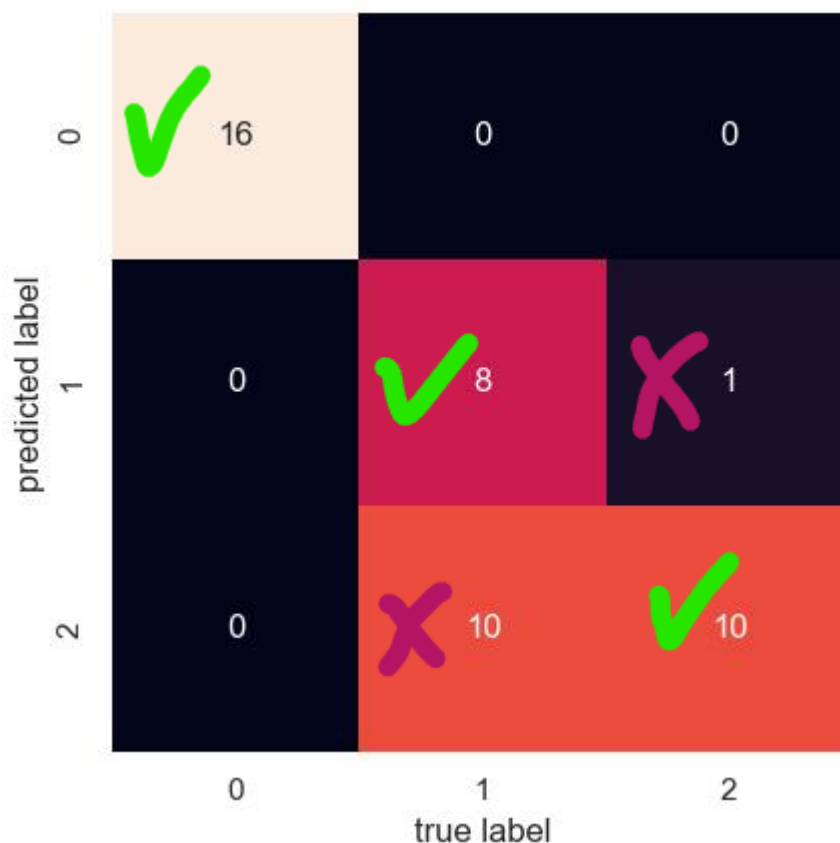


Рис 22. Візуалізація

Можна зробити висновки, що модель передбачує клас 0 ідеально, але плутається з 1 та 2 класами.

Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують

Коефіцієнт Коєна Каппа визначає наскільки збіг між двома оцінювачами перевищує те, що можна було б очікувати лише за умови випадковості, але цей коефіцієнт не говорить чи ці оцінки є правдивими / вірними. Можливі значення $[-1;1]$. У нашому випадку, це значення $= 0.6431$, що вказує на помірну або значну згоду між двома оцінювачами.

Коефіцієнт кореляції Метьюза оцінює або виміряє різницю між прогнозованими даними і фактичними значеннями. Можливі значення $[-1;1]$ У нашому випадку, ця різниця $= 0.6831$, що близько до 1, тому можна зробити висновок, що узгодження між передбачуваними даними та актуальними на хорошому рівні:

- $+1$ is the best agreement between the predicted and actual values.
- 0 is no agreement. Meaning, prediction is random according to the actuals

Висновок: У ході цієї лабораторної роботи я набув теоретичних і практичних навичок з дослідження різних методів класифікації даних, а також навчився порівнювати їх за допомогою мови програмування Python і спеціалізованих бібліотек.