Лабораторна робота 15.

У лабораторній роботі 13 була розглянута задача класифікації об’єктів на прикладі класифікації квіток ірису.
Для нароблення практичних навичок запропоновано, виконати за принципом «роби, як я» навчання та тестування алгоритму класифікації для іншого датасету – вин різник марок.
У якості моделі алгоритму навчання використана модель “k-найближчих сусідів». Процес навчання полягав у наступному. Навчальна вибірка поділена на дві частини. Одна частина використана для навчання алгоритму, а друга для тестування навченого алгоритму. Якість навчання оцінювалася за результатами тестування, оптимізація якості - шляхом підбору значення параметру k.

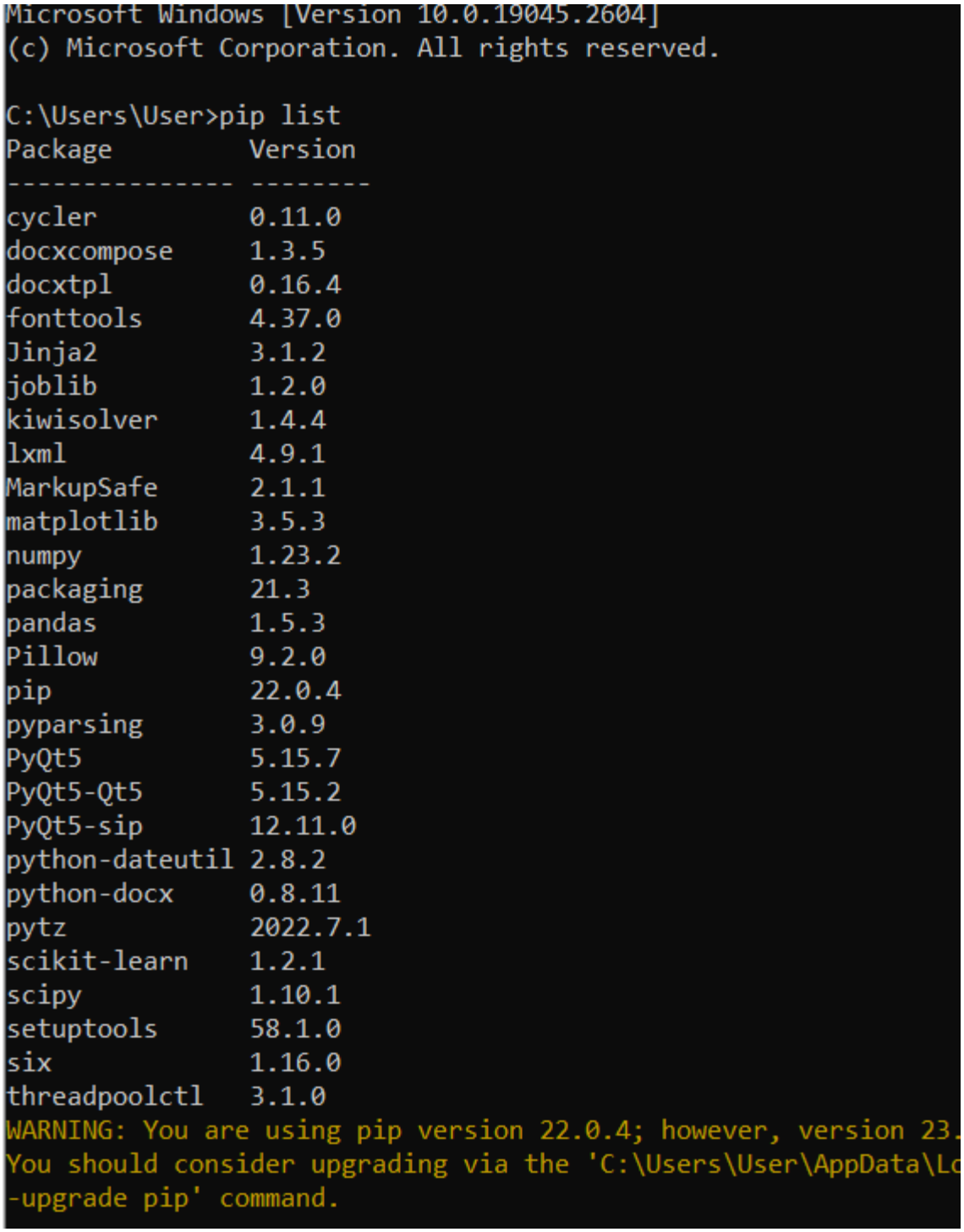
Дана робота узагальнює роботу 13 і за структурою та змістом фактично є невеликим проектом із вирішення задачі класифікації на даних, наданих замовником.
Мета роботи: отримати навички виконання проекту вирішення задачі класифікації на даних, наданих замовником.

Проект полягає у виконанні наступних завдань: Та відображення у звіті

1. Перевірка наявності та встановлення за необхідності ПЗ, а саме, платформі Python та бібліотек з набору SciPy.
2. Завантаження датасету.
3. Аналіз датасету.
4. Візуалізація даних.
5. Вибір моделей алгоритмів для вирішення поставленої задачі.
6. Навчання та вибір оптимального алгитму
7. Застосування навченого алгоритму для вирішення поставленої задачі.

1. **Перевірка наявності та встановлення необхідного ПЗ.**

Повторення – це мати навчання. Отже, подальше не є зайвим, оскільки проект виконується у середовищі, утвореному численною кількістю складових.
Перевірка може бути виконана за менеджера **pip**



Проте, цей лістинг містить увесь інструментарій, встановлений на вашій платформі Python. Отже, має сенс виконати перевірку наявності тільки необхідного інструментарію і зробити це безпосередньо у середовищі, яким ви користуєтесь. Ми домовлялися про IDLE. Код і результати виконання можуть бути такими

Iris.py - C:\Users\User\AppData\Local\Programs\Python\Iris.py (3.10.5)

File Edit Format Run Options Window Help

```
print( 'Checking Versions and loading Libraries','\n')

#1. Python
import sys
print('Python: {}'.format(sys.version),'\n')

#2. scipy
import scipy
print('scipy: {}'.format(scipy.__version__),'\n')

#3. numpy
import numpy
print('numpy: {}'.format(numpy.__version__),'\n')

#4. matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__),'\n')

#5. pandas
import pandas
print('pandas: {}'.format(pandas.__version__),'\n')

#6. scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__),'\n')


========= RESTART: C:\Users\User\AppData\Local\Programs\Pyth
on\Iris.py =========
Checking Versions and loading Libraries

Python: 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13)
 [MSC v.1929 64 bit (AMD64)]

scipy: 1.10.1

numpy: 1.23.2

matplotlib: 3.5.3

pandas: 1.5.3

sklearn: 1.2.1
```

Якщо якісь компоненти ПЗ не встановлено, це можна зробити за допомогою менеджера **pip** (див. попередній матеріал)

2. Завантаження даних

Використаємо датасет про іриси з роботи 13, і який є хрестоматійним ”Hello world” у машинному навчанні.

Нагадаємо, що цей набір містить 150 спостережень за квітками ірису. У наборі чотири колонки, які містять вимірювання квіток у сантиметрах. П’ята колонка містить значення марок (сорт) класів, яким належить кожна квітка із набору. Таких класів, як пам’ятаєте, три.

2.1. Імпорт бібліотек. Ми впевнилися, що необхідні бібліотеки встановлені на платформі Python. Проте, немає сенсу завантажувати їх повністю. Імпортуємо тільки ті модулі, які будемо використовувати:

```
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
```

2.2. Завантаження датасету. У роботі 13 ми використовували дані для ірисів, які завантажували із навчального сховища **nglearn** .

Для збільшення навичок, у цій роботі завантажимо ці дані безпосередньо із сховища, що розташовано в інтернет

```
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)
print(dataset.shape)

print(dataset.head(20))
```

```
(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
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
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
```

3. Аналіз датасету

У підпункті 2.2. ми отримали загальне уявлення про дані – кількість та назву атрибутів та кількість об'єктів у вибірці. Для наочності показали зріз даних – значення атрибутів перших двадцяти об'єктів.

Наша мета – статистичними методами отримати більш докладну інформацію про дані

3.1. Кількість екземплярів у вибірці, середнє, мін і макс значень кожного атрибуту, а також деякі процентілі ([Що таке процентиль? (greelane.com)]).

Ці характеристики можна отримати методом describe

```
# Статистичне зведення методом describe
print(dataset.describe())
```

```
       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
```

Бачимо що всі значення числові і мають однакову шкалу (сантиметри) й аналогічні діапазони від от 0 до 8 сантиметрів. Кількість екземплярів нам була

відома наперед. Проте, у загальному випадку інформації про це може не бути, а крім того, дані можуть бути не повні і кількість значень атрибутів можуть не співпадати між собою і кількістю екземплярів.

3.2. З тієї з самою причини незайвою може бути інформація про кількість екземплярів кожного класу

```
# Розподіл значень за атрибутом 'class'
print(dataset.groupby('class').size())
```

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

4. Візуалізація даних

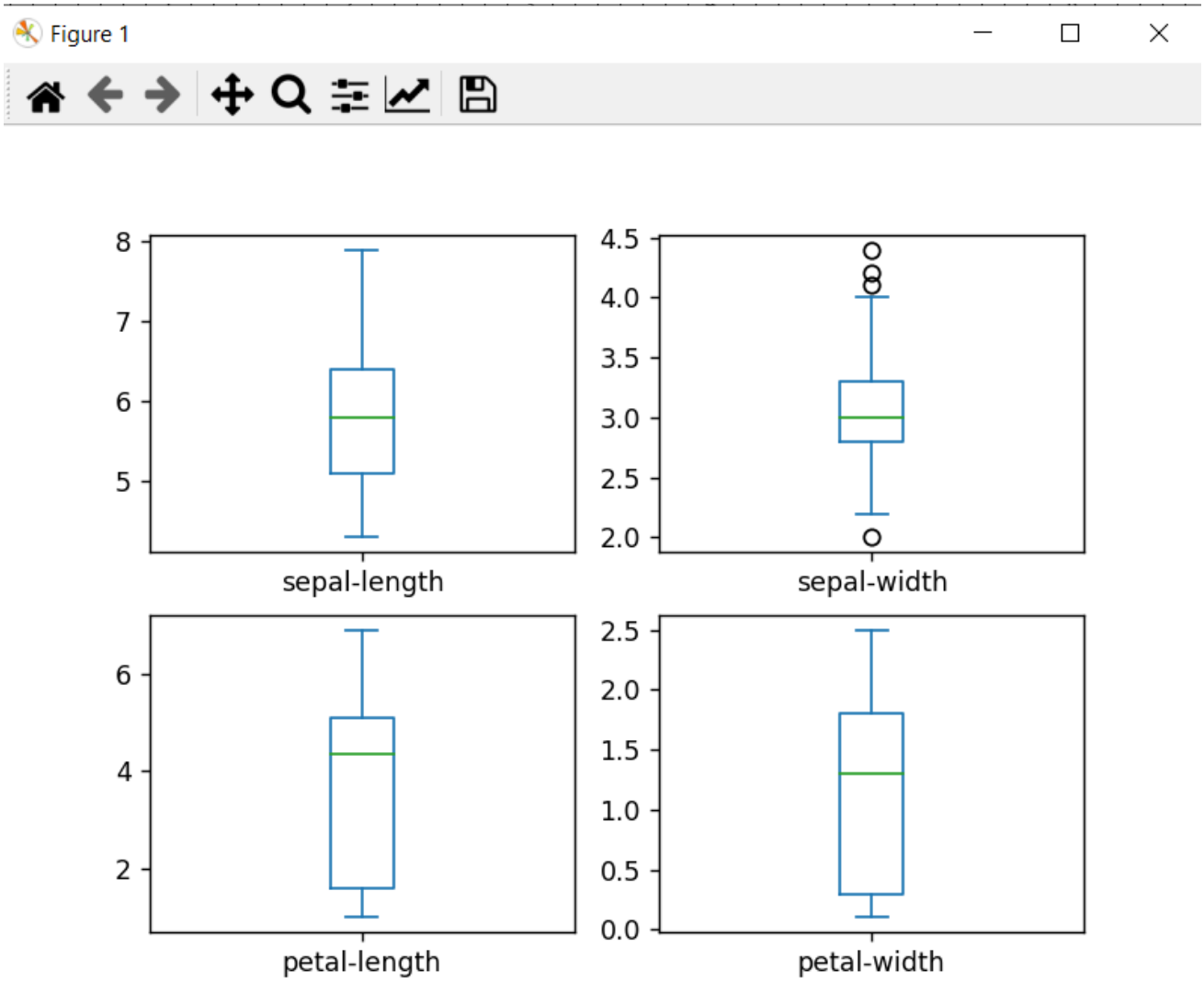
Візуалізація доповнює на розширяє уявлення про дані.
Розглянемо два типи графіків:

- Одновимірні (Univariate) графіки візуалізують характеристики та розподіл значень у межах одного атрибуту.
- Багатовимірні (Multivariate) графіки візуалізують взаємозв'язок між атрибутами.

4.1 Одновимірні графіки

Усталеним є використання діаграм розмаху (або "box and whiskers diagram"). Це досить простий спосіб візуалізувати статистичні характеристики та розподіл значень кожного з атрибутів екземплярів всієї вибірки.

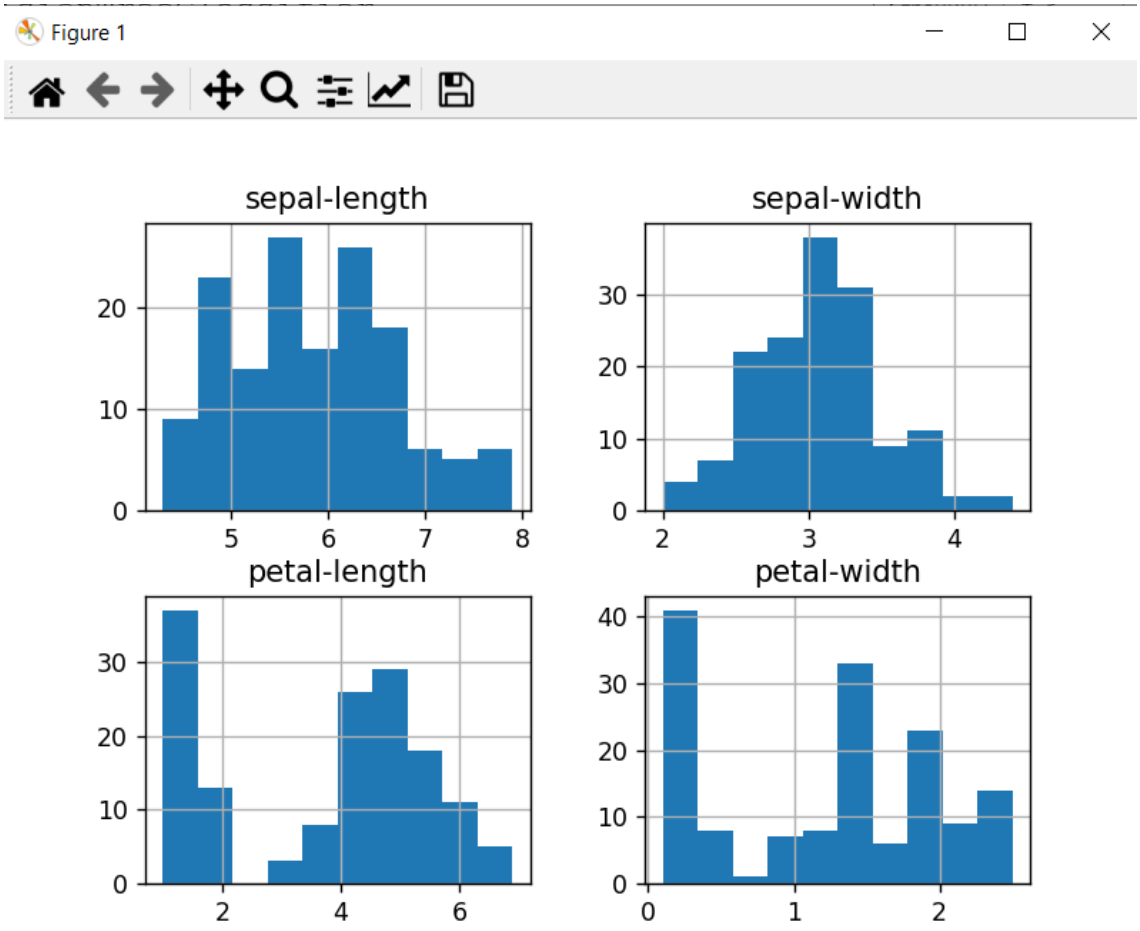
```
# Діаграма розмаху
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
pyplot.show()
```



Всі горизонтальні відрізки, що обмежують ящик та вуса відповідають числовим характеристика, показаним лістингу у п.п. 3.3. Маленькі кола – це, так звані, «викиди». Тобто значення, які не підпорядковані закономірності, яка закладена у використаний метод візуалізації даних.

Наступний код рисує гістограму частот, що дає уявлення про характер розподілу випадкових величин - значень кожного атрибуту .

```
# Гістограма розподілу значень атрибутів датасету
dataset.hist()
pyplot.show()
```



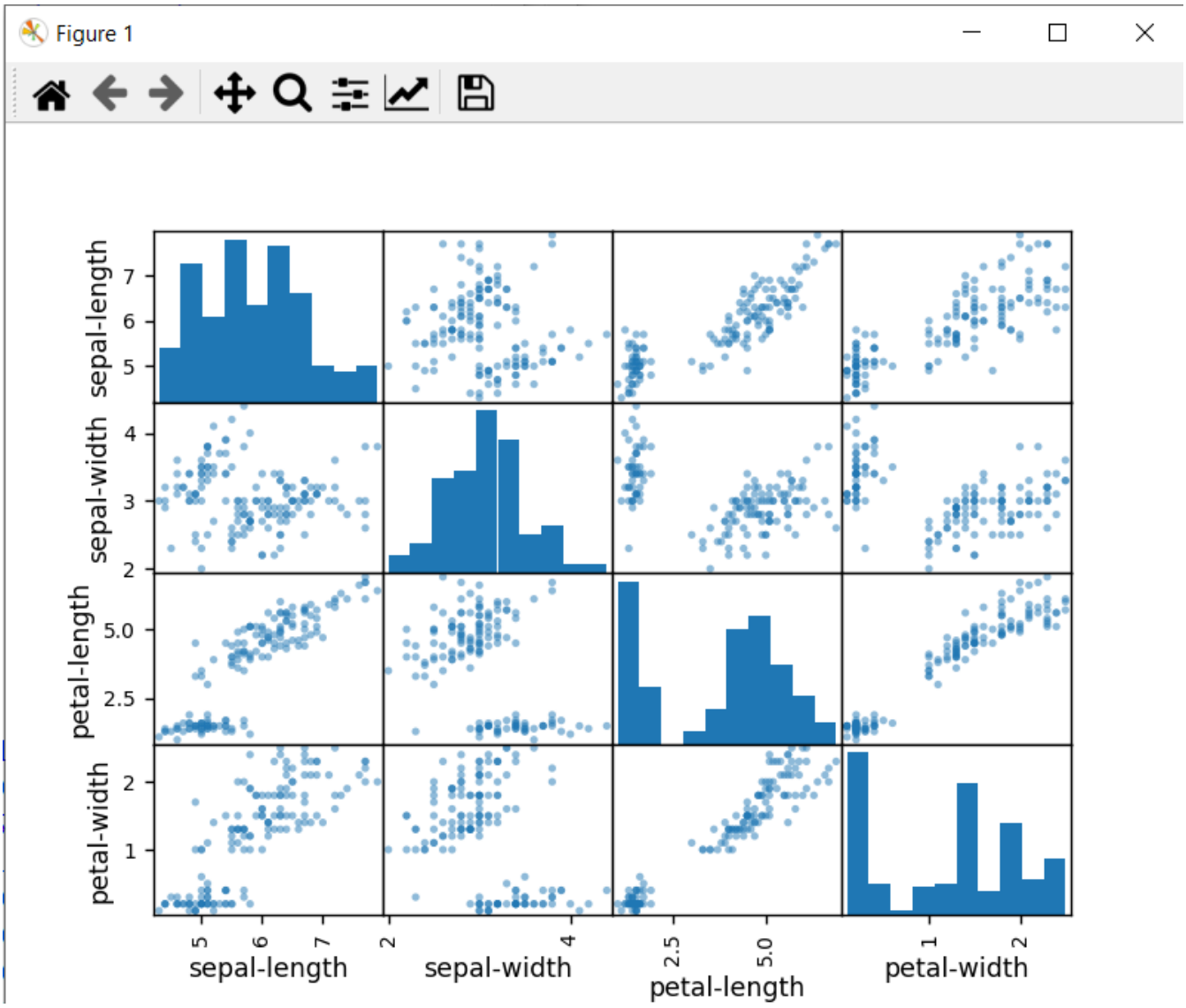
Легко бачити, що двоє з атрибутів мають розподілення, схоже на гаусове (нормальне) розподілення. Ця інформація корисна тим, дає змогу використати у навчанні алгоритми, які враховують таке припущення.
Решта два мають якесь інше, не визначене, розподілення. У цьому випадку для навчання використовується «ансамбль моделей» і вибирається оптимальна. Цей підхід і використаний у подальшому.

4.2 Багатовимірні графіки

Застосований тут метод візуалізації вже використаний у роботі 13. Нагадаємо тільки, що на листі можна показувати тільки двовимірні залежності. Отже, алгоритм рисує точки, кожна з яких відповідає екземпляру з вибірки, а її координати - парі значень атрибутів.
Такий спосіб дає уявлення про існування зв'язку між значеннями атрибутів.

Діагональ утворюють гістограми вже отримані нами. Доцільність появи їх у цьому рисунку полягає у тому, що вони, разом з графіками, демонструють наявність згрупувань, тобто класів на множині екземплярів.

```
#Матриця діаграм розподілу
scatter_matrix(dataset)
pyplot.show()
```



5. Оцінка алгоритмів

Створення алгоритмів розпізнавання на підставі деякі моделей і оцінка якості їх роботи за контрольними даними.
Вказаний процес полягає у виконанні наступних кроків:

- Ділення датасету на навчальну вибірку (застосовується для навчання алгоритму) і на тестову (контрольну) вибірку.
- Налаштування процесу 10-кратной крос-валідації на підставі навчальної вирки
- За допомогою цього процесу навчання декількох різних моделей для розпізнавання (прогнозування) належності квітки до певного сорту (класу) за результатами вимірювання її характеристик і вибір найкращої моделі.

5.1 Створення навчальної та тестової (контрольної) вибірки

Поділимо завантажений датасет на два:

- 80% даних використаємо для навчання, оцінки і вибору найкращої серед відібраних моделей алгоритму розпізнавання (класифікації).
- 20% даних, які не будемо використовувати для навчання, а використаємо у якості тестового набору для оцінки якості кожної з моделей.

Застосований підхід є усталеним. Суттєва відміна даної лабораторної роботи полягає у тому, що вона являє собою певне наближення до реальної роботи (проект):

- на навчальному наборі будемо навчати ансамбль моделей, щоб вибрати найкращу;
- на тестовому оцінимо якість розв'язування задачі класифікації ірисів.

```
# Ділення датасета на навчальну та контрольну вибірки
array = dataset.values

# Вибір перших 4-х стовбчиків
X = array[:,0:4]

# Вибір 5-го стовбчика
y = array[:,4]

# Ділення датасету
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)
```

Нагадаємо, що об'єм навчальної та тестової вибірок можна, за необхідності, змінювати значенням параметру **test_size** (робота 13).

Тепер є дані X_train и Y_train для навчання моделей і тестова (контрольна) вибірка X_validation й Y_validation для перевірки якості навчання.
*Зауважимо, що, по-суті, тестування буде виконане на матеріалі замовника (**чому?**). Такий процес тестування називається **валідацією**.*
*Тип даних датасету — "словник". Проте ми використовуємо інструментарій бібліотеки **numpy** для оброблення числових даних. Як це розуміти?*
(ми використали зріз словника, який є числовим масивом NumPy).
5.2 Навчання моделі алгоритму

Для оцінювання моделей будемо використовувати ту ж саму метрику, що і в роботі 13. Це відсоток кількості правильно передбачених екземплярів до загальної кількості екземплярів у наборі даних (наприклад, точність 95%).
Така метрика так і називається - «model acccuracy» моделей.
Оцінку отримаємо для кожної моделі.

Для навчання **кожної** з моделей використаємо процес 10-кратної крос-валідації:
a) Дані з **train-test** випадково ділитися на 10 частин – навчальний набір із 9 частин й «запасна» частина.
b) Модель навчається на навчальному наборі й оцінюється точність за вказаною метрикою.
c) Ідемо на п A), якщо кількість прогонів менша за десять (у межах 10-кратної крос-валідації).
d) Знаходимо середнє для оцінок точності, отриманих на кожному прогоні моделі.

За бажанням в інтернет легко знайти матеріал щодо цієї техніки.

5.3 Ансамбль моделей

У роботі 13 використана одна модель алгоритму розпізнавання k-найближчих сусідів (KNN).

Як правило, ми не знаємо, які моделі алгоритмів найкращі для наданого набору даних. Не вникаючи до суті, утворимо набір (ансамбль) алгоритмів і виберемо найкращій:

- Логістична регресія (LR)
- Лінійний дискримінантний аналіз (LDA)
- Метод k-найближчих сусідів (KNN)
- Класифікація й регресія за допомогою дерев (CART)
- Наївний байєсівський класифікатор (NB)
- Метод опорних векторів (SVM)

Залежність між ознаками може бути як лінійною, так і не лінійною. Отже ансамбль включає лінійні (LR и LDA) і нелінійні (KNN, CART, NB и SVM) моделі алгоритмів.

Код:

```
# Завантаження моделей алгоритмів
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')))

# оцінка точності модель на каждой итерации
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()))
```

Результат:

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.040825)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

5.4 Вибір найкращої моделі

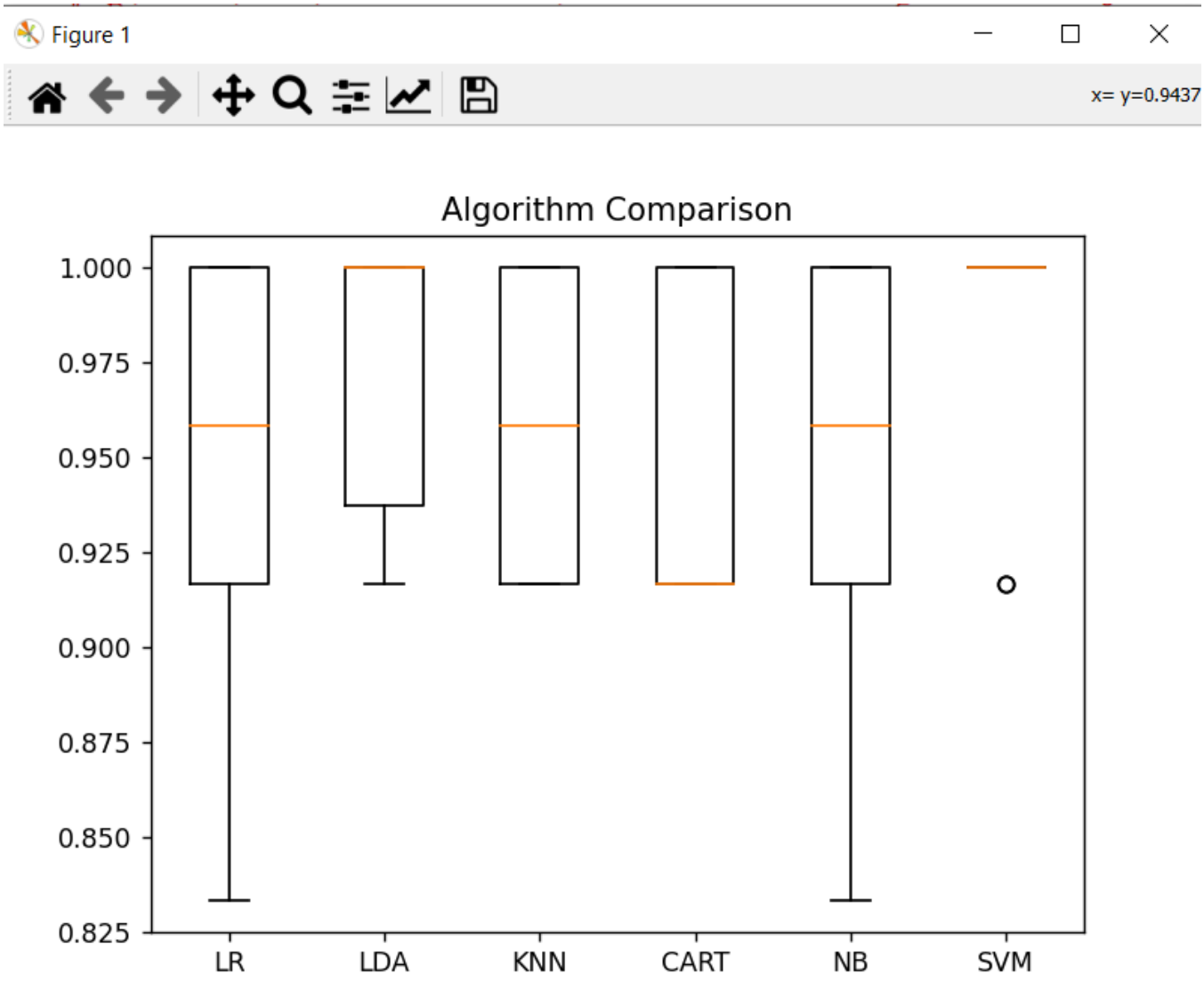
Є шість навчених моделі й оцінка точності кожної з них (див. вище).
Порівняємо їх.

*Зауважимо, що результати, отримані **вами**, можуть трохи відрізнятися від наведених. **Чому?***

Висновок з наведених результатів, що найкращім є метод опорних векторів (SVM), оскільки його точність найбільша, біля 98% правильно класифікованих квіток із тренувального набору. Непоганою альтернативою може бути лінійний алгоритм LDA, але середнє квадратичне відхилення трохи більше.
Результати навчання ансамблю можна візуалізувати тими самими ящиками з вусами.
Червоний відрізок – середнє арифметичне значень точності всіх прогонів (10) моделі. Ящик та вуса – процентілі, які показують «розмах» значень точності на кожному прогоні.
Точність моделі 98% відрізняється від 100%, чому?
До речі, наявність викиду, може бути вагомим аргументом віддати перевагу, все ж таки, лінійному алгоритму LDA. Проте, для остаточного висновку необхідні додаткові дослідження за межами цього навчального матеріалу 😊.
Код візуалізації:

```
# Візуалізація результатів навчання ансамблю моделей
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
```

Результат



6. Прогнозування даних

Задача класифікації полягає у прогнозуванні (розпізнаванні) належності даного об’єкту одному з класів об’єктів, на яких навчалися алгоритми.

6.1. Валідація алгоритму SVM.

Найкращій результат з ансамблю моделей показ алгоритм SVM.

У процесі його навчання не використовувалася контрольна (тестова) вибірка - 20% даних з датасету. Їх ми й використаємо для валідації алгоритму. Тобто, вважаємо, що це дані, які надав замовник для оцінки результатів виконання проекту.

Доцільно зробити так:
Отримати прогноз і його точність на контрольній (тестовій) вибірці.
Отримати прогноз і його точність на всьому тренувальному наборі.
Порівняти ці точності. Це дасть нам змогу **оцінити точність якість прогнозу при валідації алгоритму** .
Код

```
# Прогноз на тренувальній та контрольній вибірках
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

# Оцінка прогнозу
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

Результат

```
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

6.1. Оцінювання якості прогнозу

Дані замовника містять розміри 30-ти об’єктів (20%), що не використовувалися при навчанні.
Точність прогнозу на даних замовника 97%. Це відповідає точності на тренувальному наборі 98%.
Матриця кількості помилок дає уявлення про одну помилку при розпізнаванні належності квітки сорту **iris-verginika**.
Остаточний звіт містить загальну точність прогнозу, точність прогнозу за класами і середню зважену точність за класам. Останнє відповідає загальній точності, що дає підставу для висновку про придатну якість прогнозу.

Завдання

1. Поновити наведений код і отримати аналогічні результати.
2. Розв’язати задачу класифікації за даними про їх ознаки. Зробити для wine!