Лабораторна робота №4

Тема: Класифікація методом k найближчих сусідів і набір даних Digits, частина 1

Щоб пошта оброблялася ефективно, а кожен лист передавалося за правильною адресою, комп'ютери поштової служби повинні сканувати

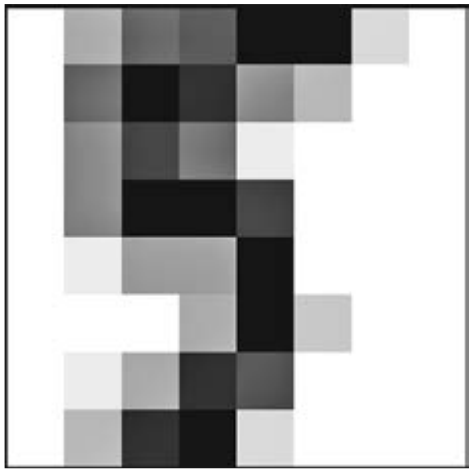
рукописні імена, адреси та поштові індекси, розпізнаючи цифри і букви. Завдяки потужним бібліотекам, таким як scikit-learn, навіть початківець програміст здатний впоратися з подібним завданням з області машинного навчання.

Завдання класифікації

задачу класифікації в області машинного навчання з учителем, де потрібно спрогнозувати класс, до якого відноситься зразок. Наприклад, якщо у вас є зображення собак і кішок, то кожне зображення має класифікуватися як «собака» або «кішка». Подібна задача називається бінарною, оскільки в ній задіяні всього два класи.

Скористаємося набором даних Digits, що входять в поставку scikit-learn. Набір складається з зображень 8 × 8 пікселів і представляє тисячі сімсот дев'яносто сім рукописних цифр (від 0 до 9). Потрібно визначити, яку цифру представляє зображення. Так як існує 10 можливих цифр (класів), дана задача є задачею множинної класифікації. Для навчання моделі використовуються помічені дані - клас кожної цифри відомий заздалегідь. В цій лабораторній роботі для розпізнавання рукописних цифр буде застосований один з найпростіших алгоритмів класифікації - метод k найближчих сусідів (k-NN).

Представимо візуалізацію цифри 5 у низькій роздільній здатності була отримана в результаті виведення однієї цифри в вигляді матриці 8 × 8, зображення засобами Matplotlib:



Дослідники створили зображення цього набору даних на основі бази даних MNIST з десятками тисяч зображень 32 × 32 пікселя, отриманих на початку 1990-х. З сучасними камерами і сканерами високої роздільної здатності такі зображення можна записати з більш високою якістю.

Мета роботи:

Навчитись реалізовувати основні етапи машинного навчання:

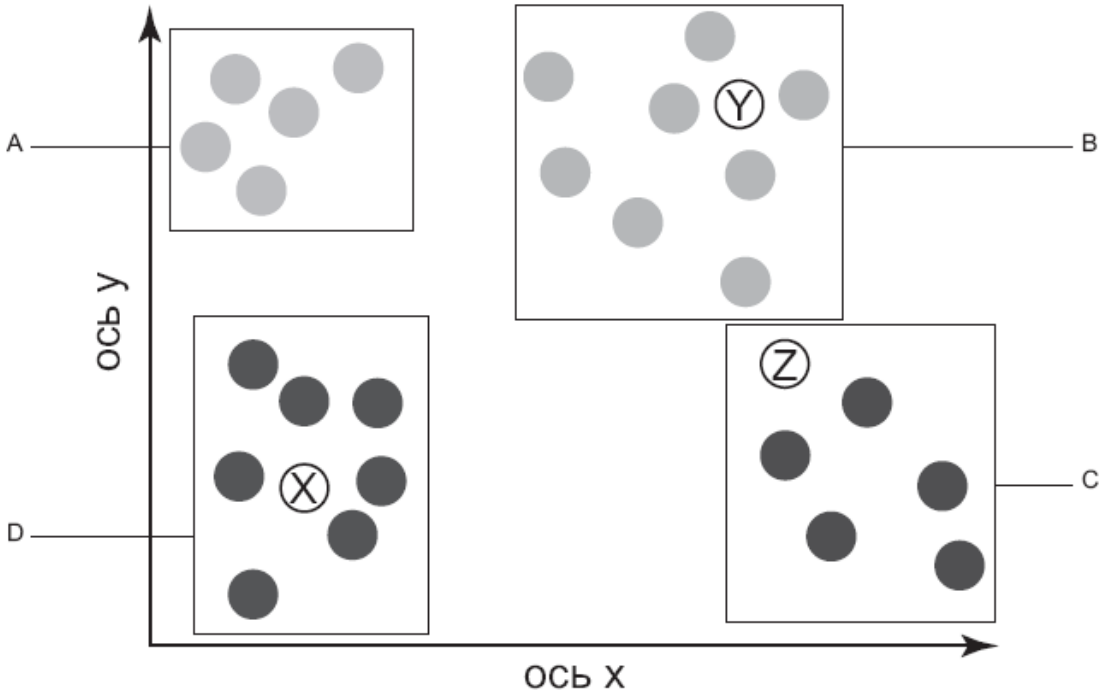
- Вибір даних для навчання моделі.
- Завантаження та аналіз даних.
- Розбиття даних для навчання і тестування.
- Вибір і побудова моделі.
- Навчання моделі.
- Формування прогнозів.
- проведемо оцінку результатів;
- налаштуємо параметри моделі;
- опрацюємо кілька класифікаційних моделей для вибору найкращої моделі (-ей).

Для візуалізації даних використати бібліотеки Matplotlib і Seaborn, тому IPython слід запустити з підтримкою Matplotlib:

```
ipython --matplotlib
```

Алгоритм k найближчих сусідів

Scikit-learn підтримує багато алгоритмів класифікації, включаючи найпростіший алгоритм k найближчих сусідів (k-NN). Цей алгоритм намагається спрогнозувати клас тестового зразка, аналізуючи k навчальних зразків, розташованих найближче (по відстані) до тестового зразком. Для прикладу візьмемо наступну діаграму, на якій заповнені точки представляють чотири класи - A, B, C і D. У контексті нашого обговорення ці літери будуть використовуватися як імена класів:



Потрібно спрогнозувати класи, до яких належать нові зразки X, Y і Z. Будемо вважати, що прогнози повинні формуватися за трьома найближчим сусідам кожного зразка - k дорівнює 3 в алгоритмі k найближчих сусідів:

- Всі три найближчих сусіда зразка X є точками класу D, тому модель прогнозує, що X відноситься до класу D.
- Всі три найближчих сусіда зразка Y є точками класу B, тому модель прогнозує, що Y відноситься до класу B.
- Для Z вибір не очевидний, тому що зразок перебуває між точками B і C. З трьох найближчих сусідів одна належить класу B, а два -

класу C. У алгоритмі k найближчих сусідів перемагає клас з більшістю «голосів». Через двох голосів C проти одного голосу B ми прогнозуємо, що Z відноситься до класу C. Вибір непарного значення k в алгоритмі k-NN попереджує «нічиї» і гарантує, що кількість голосів ніколи не буде рівних.

Гіперпараметри і настройка гіперпараметров

В області машинного навчання модель реалізує алгоритм машинного навчання. У термінології scikit-learn моделі називаються оцінювачами. Існують два типи параметрів машинного навчання:

- обчислювані оцінювачем в ході свого навчання на підставі наданих вами даних;
- задаються заздалегідь при створенні об'єкта оцінювача scikit-learn, що представляє Модель.

Параметри, що задаються заздалегідь, називаються гіперпараметрамі. В алгоритмі k найближчих сусідів k є гіперпараметром.

Для простоти ми використовуємо значення гіперпараметров за замовчуванням для scikit-learn. У цьому дослідженні з області машинного навчання бажано поекспериментувати з різними значеннями k для отримання найкращих можливих моделей для ваших досліджень. Цей процес називається настроюванням гіперпараметров. В останній частині лабораторної роботи (пункт 10) виконаємо налаштування гіперпараметров для вибору значення k, яке дозволяє алгоритму k найближчих сусідів видати кращі прогнози для набору даних Digits. Scikit-learn також містить засоби автоматичної настройки гіперпараметров.

Завантаження набору даних

Функція load_digits з модуля sklearn.datasets повертає об'єкт scikit-learn Bunch, що містить дані цифр і інформацію про набір даних Digits (так звані метадані):

```
In [1]: from sklearn.datasets import load_digits

In [2]: digits = load_digits()
```

Bunch є підклас dict, що містить додаткові атрибути

для взаємодії з набором даних.

Набір даних Digits, що входить в поставку scikit-learn, є підмножиною набору даних рукописних цифр UCI (Каліфорнійський університет в Ірвайні) ML:

http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digit

Вихідний набір даних UCI містить 5620 зразків - 3823 для навчання і тисяча сімсот дев'яносто сім для тестування. Версія набору даних, що поставляється з scikit-learn, містить тільки тисячі сімсот дев'яносто сім тестових зразків. Атрибут DESCR об'єкта Bunch містить

опис набору даних. Відповідно до опису набору даних Digits1, кожен зразок містить 64 ознаки (Number of Attributes), що представляють

зображення 8 × 8 зі значеннями пікселів в діапазоні 0-16 (Attribute Information). Набір даних не містить відсутніх значень (Missing Attribute Values). Створюється враження, що 64 ознаки - це багато, але необхідно мати на увазі, що реальні набори даних іноді містять сотні, тисячі і навіть мільйони ознак.

```
In [3]: print(digits.DESCR)
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 5620
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range
     0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
http://archive.ics.uci.edu/ml/datasets/
   Optical+Recognition+of+Handwritten+Digits
...
```

Перевірка атрибутів data і target Атрибути data і target об'єкта Bunch представляють собою масиви NumPy:

- Масив data містить 1797 зразка (зображення цифр), кожен з яких несе 64 ознаки зі значеннями в діапазоні 0-16, що представляють інтенсивності пікселів. З Matplotlib можна візуалізувати інтенсивності в відтінках сірого від білого (0) до чорного (16):

- Масив target містить мітки зображень, тобто класи, які вказують,

яку цифру представляє кожне зображення. Масив називається target, тому що при прогнозуванні ви прагнете «влучити в ціль» з вибором значень. Щоб побачити мітки зразків в наборі даних, виведемо

значення target кожного 100-го зразка:

```
In [4]: digits.target[::100]
Out[4]: array([0, 4, 1, 7, 4, 8, 2, 2, 4, 4, 1, 9, 7, 3, 2, 1, 2, 5])
```

Кількість зразків і ознак (на один зразок) підтверджується за допомогою атрибута shape масиву data, який показує, що набір даних складається з тисячі сімсот дев'яносто-сім рядків (зразків) і 64 стовпців (ознак):

```
In [5]: digits.data.shape
Out[5]: (1797, 64)
```

Розміри масиву target підтверджують, що кількість цільових значень відповідає кількості зразків:

```
In [6]: digits.target.shape
Out[6]: (1797,)
```

Завдання:

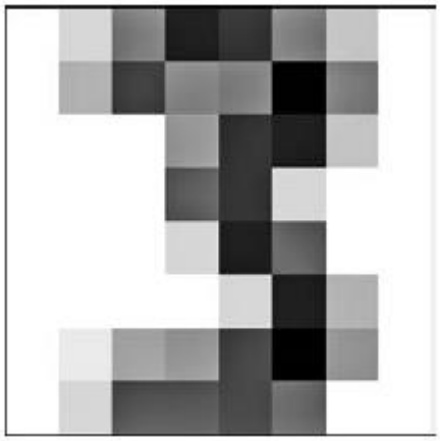
1. Для дослідження даних, візуалізуйте їх. Виведіть зображення перших 24 і 36 цифр з набору

Всі зображення двовимірні - вони володіють шириною і висотою в пікселах. Об'єкт Bunch, що повертається load_digits, містить атрибут images - масив, кожен елемент якого являє собою двовимірний масив 8 × 8 з інтенсивностями пікселів зображення цифри. Хоча в початковому наборі даних кожен піксель представлений цілочисельним значенням в діапазоні 0- 16, scikit-learn зберігає ці значення в вигляді значень з плаваючою точкою (тип NumPy float64). Наприклад, двовимірний масив, що представляє зображення зразка з індексом 13, виглядає так:

```
In [7]: digits.images[13]
Out[7]:
array([[ 0.,  2.,  9., 15., 14.,  9.,  3.,  0.],
       [ 0.,  4., 13.,  8.,  9., 16.,  8.,  0.],
```

```
[ 0.,  0.,  0.,  6., 14., 15.,  3.,  0.],
[ 0.,  0.,  0., 11., 14.,  2.,  0.,  0.],
[ 0.,  0.,  0.,  2., 15., 11.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  2., 15.,  4.,  0.],
[ 0.,  1.,  5.,  6., 13., 16.,  6.,  0.],
[ 0.,  2., 12., 12., 13., 11.,  0.,  0.]])
```

Нижче показано зображення, представлене цим двовимірним масивом - незабаром ми наведемо код виведення цього зображення:



Підготовка даних для використання з scikit-learn Алгоритми машинного навчання scikit-learn вимагають, щоб зразки зберігалися в двовимірному масиві значень з плаваючою точкою (або колекції, схожої з двовимірним масивом, наприклад списком списків або колекцією pandas DataFrame):

- кожен рядок являє один зразок;
- кожен стовпець заданої рядки представляє одна ознака цього зразка.

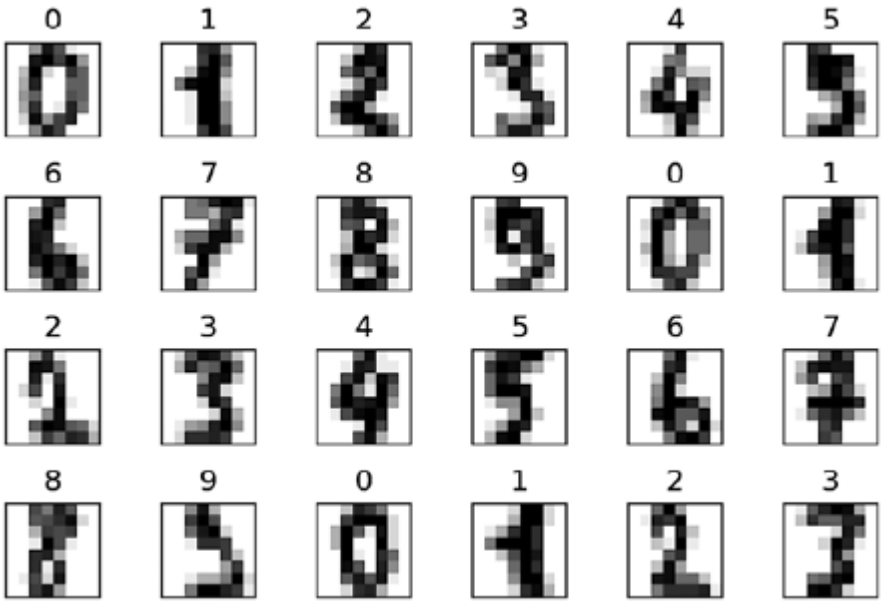
Для представлення кожного зразка у вигляді одного рядка даних багатовимірні дані (наприклад, двовимірний масив image з фрагмента [7]) повинні бути перетворені в одновимірний масив.

Якщо ви працюєте з даними, які містять категорійні ознаки (зазвичай представлені у вигляді рядків - скажімо, 'spam' і 'not-spam'), то вам також доведеться провести попередню обробку цих ознак і перетворити їх в числові значення (так званий прямий унітарне кодування буде розглядатися в наступному розділі). Модуль sklearn.preprocessing бібліотеки Scikit-learn надає функціональність для перетворення категорійних даних в числові. Набір даних Digits не містить категорійних ознак.

Для вашої зручності функція load_digits повертає попередньо оброблені дані, готові до машинного навчання. Набір даних Digits є числовим, тому load_digits просто згладжує двовимірний масив в одновимірний масив. Наприклад, масив 8 × 8 digits.images [13] з фрагмента [7] відповідає масиву 1 × 64 digits.data [13] такого вигляду:

```
In [8]: digits.data[13]
Out[8]:
array([ 0.,  2.,  9., 15., 14.,  9.,  3.,  0.,  0.,  4., 13.,  8.,  9.,
       16.,  8.,  0.,  0.,  0.,  0.,  6., 14., 15.,  3.,  0.,  0.,  0.,
        0., 11., 14.,  2.,  0.,  0.,  0.,  0.,  0.,  2., 15., 11.,  0.,
        0.,  0.,  0.,  0.,  0.,  2., 15.,  4.,  0.,  0.,  1.,  5.,  6.,
       13., 16.,  6.,  0.,  0.,  2., 12., 12., 13., 11.,  0.,  0.])
```

У цьому одновимірному масиві перші вісім елементів містять елементи рядка 0 двовимірного масиву, наступні вісім елементів - елементи рядка 1 двовимірного масиву, і т. Д.




```
In [9]: import matplotlib.pyplot as plt

In [10]: figure, axes = plt.subplots(nrows=4, ncols=6, figsize=(6, 4))

In [11]: for item in zip(axes.ravel(), digits.images, digits.target):
             axes, image, target = item
             axes.imshow(image, cmap=plt.cm.gray_r)
             axes.set_xticks([])
             axes.set_yticks([])

         axes.set_title(target)
     plt.tight_layout()
```

2. Розбийте дані на навчальні та тестові, за замовчуванням train_test_split резервує 75% даних для навчання і 25% для тестування, змініть це.

Спочатку дані розбиваються на два поднабора: навчальний і тестовий. Функція train_test_split з модуля sklearn.model_selection здійснює випадкову перестановку даних, а потім розбиває зразки в масиві data і цільові значення в масиві target на навчальний і тестовий набір. Це гарантує, що навчальний і тестовий набори володіють схожими характеристиками.

Випадкова перестановка і розбиття виконуються для вашої зручності об'єктом ShuffleSplit з модуля sklearn.model_selection. функція

train_test_split повертає кортеж з чотирьох елементів, в якому два перших елемента містять зразки, розділені на навчальний і тестовий набір, а два останніх - відповідні цільові значення, також розділені

на навчальний і тестовий набір. За загальноприйнятим угодами буква верхнього регістру X використовується для представлення зразків, а буква y нижнього регістру - для подання цільових значень:

```
In [12]: from sklearn.model_selection import train_test_split

In [13]: X_train, X_test, y_train, y_test = train_test_split(
             digits.data, digits.target, random_state=11)
```

```
In [14]: X_train.shape
Out[14]: (1347, 64)

In [15]: X_test.shape
Out[15]: (450, 64)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    digits.data, digits.target, random_state=11, test_size=0.20)
```

Щоб використовувати інше співвідношення, можна задати розміри тестового і навчального набору за допомогою ключових аргументів test_size і train_size функції train_test_split. Використовуйте значення з плаваючою точкою в діапазоні від 0.0 до 1.0 для визначення відсоткової частки кожного набору в даних. Цілочисельні значення задають точну кількість зразків. Якщо один з цих ключових аргументів задається при виклику, то другий обчислюється автоматично. Остання команда повідомляє, що 20% даних призначені для тестування, тому значення train_size обчислюється рівним 0.80.

3. Створити та навчити модель

Оцінювач KNeighborsClassifier (модуль sklearn.neighbors) реалізує алгоритм k найближчих сусідів. Спочатку створюється об'єкт оцінювача KneighborsClassifier. Вам залишається просто викликати методи цього об'єкту

```
In [16]: from sklearn.neighbors import KNeighborsClassifier

In [17]: knn = KNeighborsClassifier()
```

Викликаємо метод fit об'єкта KNeighborsClassifier, який завантажує навчальний набір зразків (X_train) і навчальний набір цільових значень (y_train) в оцінювача:

```
In [18]: knn.fit(X=X_train, y=y_train)
Out[18]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

Значення n_neighbors відповідає k в алгоритмі k найближчих сусідів. За замовчуванням KNeighborsClassifier шукає п'ятьох найближчих сусідів для побудови своїх прогнозів. Для простоти використати оцінки оцінювача за замовчуванням. Для KNeighborsClassifier вони описані за адресою: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

4. Виконайте прогнозування класів

```
In [19]: predicted = knn.predict(X=X_test)

In [20]: expected = y_test
```

5. Порівняйте прогнозовані цифри з очікуваними для перших 20, 24, 36 тестових зразків:

```
In [21]: predicted[:20]
Out[21]: array([0, 4, 9, 9, 3, 1, 4, 1, 5, 0, 4, 9, 4, 1, 5, 3, 3, 8, 5, 6])

In [22]: expected[:20]
Out[22]: array([0, 4, 9, 9, 3, 1, 4, 1, 5, 0, 4, 9, 4, 1, 5, 3, 3, 8, 3, 6])
```

6. Поясніть результат, застосуйте метрики точності моделі.

6.1 Метод score оцінювача

Кожен оцінювач містить метод score, який повертає оцінку результатів,

показаних з тестовими даними, переданими в аргументах**.**

```
In [25]: print(f'{knn.score(X_test, y_test):.2%}')
97.78%
```

6.2 Матриця невідповідностей.

містить інформацію про правильно і неправильно прогнозованих значеннях (також званих влученнями і промахами) для заданого класу.

```
In [26]: from sklearn.metrics import confusion_matrix

In [27]: confusion = confusion_matrix(y_true=expected, y_pred=predicted)

In [28]: confusion
Out[28]:
array([[45,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 45,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 54,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 42,  0,  1,  0,  1,  0,  0],
       [ 0,  0,  0,  0, 49,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 38,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 42,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 45,  0,  0],
       [ 0,  1,  1,  2,  0,  0,  0,  0, 39,  1],
       [ 0,  0,  0,  0,  1,  0,  0,  0,  1, 41]])
```

7. Виведіть звіт класифікації

Модуль sklearn.metrics також надає функцію classification_report, яка виводить таблицю метрик классіфікації1, заснованих на очікуваних і прогнозованих значеннях:

```
In [29]: from sklearn.metrics import classification_report

In [30]: names = [str(digit) for digit in digits.target_names]

In [31]: print(classification_report(expected, predicted,
    ...:       target_names=names))
    ...:
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	45
1	0.98	1.00	0.99	45
2	0.98	1.00	0.99	54
3	0.95	0.95	0.95	44
4	0.98	0.98	0.98	50
5	0.97	1.00	0.99	38
6	1.00	1.00	1.00	42
7	0.96	1.00	0.98	45
8	0.97	0.89	0.93	44
9	0.98	0.95	0.96	43
micro avg	0.98	0.98	0.98	450
macro avg	0.98	0.98	0.98	450
weighted avg	0.98	0.98	0.98	450

8. Використайте декілька моделей KNeighborsClassifier, SVC и GaussianNB для пошуку найкращої

```
from sklearn.svm import SVC

from sklearn.naive_bayes import GaussianNB
```

9. Налаштуйте гіперпараметр K в KNeighborsClassifier

Щоб визначити краще значення k в алгоритмі k-NN, поекспериментуйте з різними значеннями k і порівняйте ефективність оцінювача в будь-якому вигляді.

10. Зробити звіт про роботу, який включає:

а. Титульна сторінка з інформацією про виконавця, темою та номером лабораторної роботи,

б. Постановку завдання

в. Скріни коду та скріни результату виконання з коментарями

г. Висновок

11. Надіслати звіт(.docx або .doc) на @zeit_13 (Telegram) або aonesterukr@gmail.com (e-mail). В повідомленні обов'язково вказати Ваше ПІБ, групу, назву предмету (скорочено)

12. Записати захист роботи на відео та надіслати його разом зі звітом на вище вказані адреси. Під час захисту роботи показати роботу програмного коду! (не звіт) та пояснити свої дії.