

Міністерство освіти і науки України  
Національний технічний університет України «Київський політехнічний інститут  
імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №4 з дисципліни  
«Прикладні задачі машинного навчання»

«Класифікація методом k найближчих сусідів  
і набір даних Digits, частина 1»

Виконав:

ПІ-13 Ал Хадам Мурат Реззович

(шифр, прізвище, ім'я, по батькові)

Перевірив:

Нестерук Андрій Олександрович

(прізвище, ім'я, по батькові)

Київ 2023

### **Постановка завдання:**

- 1. Для дослідження даних, візуалізуйте їх. Виведіть зображення перших 24 і 36 цифр з набору.**
- 2. Розбийте дані на навчальні та тестові, за замовчуванням `train_test_split` резервує 75% даних для навчання і 25% для тестування, змініть це.**
- 3. Створити та навчити модель.**
- 4. Виконайте прогнозування класів.**
- 5. Порівняйте прогнозовані цифри з очікуваними для перших 20, 24, 36 тестових зразків.**
- 6. Поясніть результат, застосуйте метрики точності моделі.**
- 7. Виведіть звіт класифікації**
- 8. Використайте декілька моделей `KNeighborsClassifier`, `SVC` і `GaussianNB` для пошуку найкращої.**
- 9. Налаштуйте гіперпараметр `K` в `KneighborsClassifier`.**

## Виконання

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

1. Імпортуємо необхідні бібліотеки та завантажимо дані

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

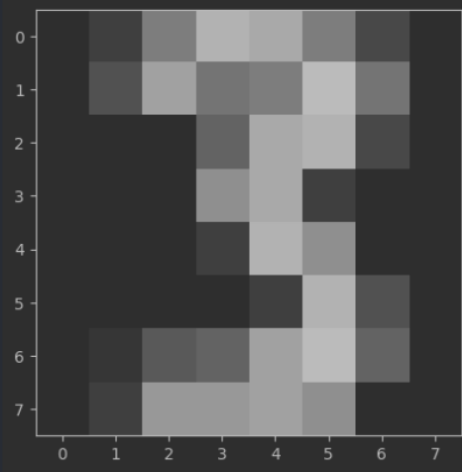
```
In 142 1 from sklearn.datasets import load_digits
2
3 digits = load_digits()
4 image_13 = digits.data[13].reshape(8, 8)
5
6 print(image_13)
7 print(digits.target[13])
```

```
[[ 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.]]
3
```

2. Зображимо картинку репрезентації по піксельному розподілу.

```
In 143 1 import matplotlib.pyplot as plt
2
3 plt.imshow(image_13, cmap=plt.cm.gray_r)
```

```
Out 143 <matplotlib.image.AxesImage at 0x218f2b16d70>
```

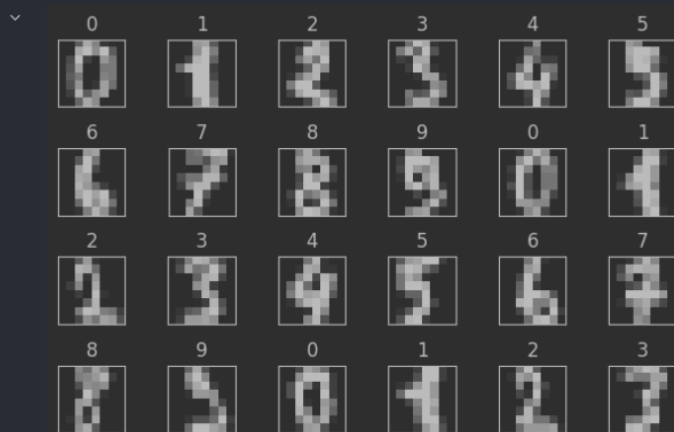


3. Вивід перших 24, 36 чисел

```

In 144 1 def display_digits(num: int):
2       figure, axes = plt.subplots(nrows=num//6, ncols=6, figsize=(6, num//6))
3
4       for item in zip(axes.ravel(), digits.images, digits.target):
5           axes, image, target = item
6           axes.imshow(image, cmap=plt.cm.gray_r)
7
8           axes.set_xticks([])
9           axes.set_yticks([])
10          axes.set_title(target)
11      plt.tight_layout()
12
13      display_digits(24)

```



```

In 145 1 display_digits(36)

```



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

```
In 146 1 from sklearn.model_selection import train_test_split
2
3 X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target, random_state=11)
4
5 print(X_train.shape)
6 print(X_test.shape)

(1347, 64)
(450, 64)
```

### 3. Створити та навчити модель

```
In 147 1 from sklearn.neighbors import KNeighborsClassifier
2
3 knn = KNeighborsClassifier()
4 knn.fit(X=X_train, y=y_train)
```

Out 147 ▾ KNeighborsClassifier  
KNeighborsClassifier()

### 4. Виконайте прогнозування класів

```
In 148 1 predicted = knn.predict(X=X_test)
2 expected = y_test
```

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

```
In 149 1 import pandas as pd
2
3 def is_correct_predict(num: int):
4     pred_n = predicted[:num]
5     exp_n = expected[:num]
6
7     print(f'For the {num}th first numbers:')
8     print(pd.DataFrame({'expected': exp_n, 'predicted': pred_n, 'is_correct': (pred_n == exp_n)}))
9
10 is_correct_predict(20)
11 is_correct_predict(24)
12 is_correct_predict(36)
```

For the 20th first numbers:

	expected	predicted	is_correct
0	0	0	True
1	4	4	True
2	9	9	True
3	9	9	True
4	3	3	True
5	1	1	True
6	4	4	True
7	1	1	True
8	5	5	True

## 6. Поясніть результат, застосуйте метрики точності моделі.

### 1. Метод score оцінювача

```
In 150 1 knn_score = knn.score(X_test, y_test)
      2 print(f'{knn_score:.2%}')
```

97.78%

### 2. Матриця невідповідностей

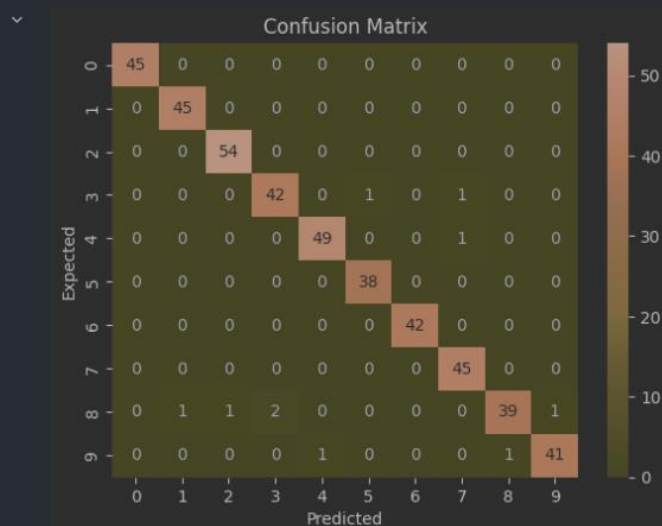
```
In 151 1 from sklearn.metrics import confusion_matrix
      2
      3 confusion = confusion_matrix(y_true=expected, y_pred=predicted)
      4 confusion
```

Out 151

	0	1	2	3	4	5	6	7	8	9
0	45	0	0	0	0	0	0	0	0	0
1	0	45	0	0	0	0	0	0	0	0
2	0	0	54	0	0	0	0	0	0	0
3	0	0	0	42	0	1	0	1	0	0
4	0	0	0	0	49	0	0	1	0	0
5	0	0	0	0	0	38	0	0	0	0
6	0	0	0	0	0	0	42	0	0	0
7	0	0	0	0	0	0	0	45	0	0
8	0	1	1	2	0	0	0	0	39	1
9	0	0	0	0	1	0	0	0	1	41

Візуалізуємо матрицю невідповідностей.

```
In 152 1 import seaborn as sns
      2
      3 sns.heatmap(confusion, annot=True, fmt='d', cmap='YlOrBr')
      4
      5 plt.xlabel('Predicted')
      6 plt.ylabel('Expected')
      7 plt.title('Confusion Matrix')
      8 plt.show()
```



## 7. Виведіть звіт класифікації

```
In 153 1 from sklearn.metrics import classification_report
      2
      3 names = [str(digit) for digit in digits.target_names]
      4 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
accuracy			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 і GaussianNB для пошуку найкращої

- SVC

```
SVC

In 154 1 from sklearn.svm import SVC
      2
      3 svc = SVC()
      4 svc.fit(X_train, y_train)
      5
      6 svc_score = svc.score(X_test, y_test)
      7 print(f'{svc_score:.2%}')

98.67%
```

- GaussianNB

```
In 155 1 from sklearn.naive_bayes import GaussianNB
      2
      3 gnb = GaussianNB()
      4 gnb.fit(X_train, y_train)
      5
      6 gnb_score = gnb.score(X_test, y_test)
      7 print(f'{gnb_score:.2%}')

      86.89%
```

## 9. Налаштуйте гіперпараметр К в KNeighborsClassifier

Порахуємо точність моделі при різних значеннях гіперпараметра:

- n\_neighbors=3

```
In 156 1 knn_test1 = KNeighborsClassifier(n_neighbors=3)
      2 knn_test1.fit(X=X_train, y=y_train)
      3
      4 knn1_score = knn_test1.score(X_test, y_test)
      5 print(f'{knn1_score:.2%}')

      98.22%
```

- n\_neighbors=1

```
In 157 1 knn_test2 = KNeighborsClassifier(n_neighbors=1)
      2 knn_test2.fit(X=X_train, y=y_train)
      3
      4 knn2_score = knn_test2.score(X_test, y_test)
      5 print(f'{knn2_score:.2%}')

      98.44%
```



- n\_neighbors=7

```
In 158 1 knn_test3 = KNeighborsClassifier(n_neighbors=7)
      2 knn_test3.fit(X=X_train, y=y_train)
      3
      4 knn3_score = knn_test3.score(X_test, y_test)
      5 print(f'{knn3_score:.2%}')

98.00%
```

- n\_neighbors=9

```
In 159 1 knn_test4 = KNeighborsClassifier(n_neighbors=7)
      2 knn_test4.fit(X=X_train, y=y_train)
      3
      4 knn4_score = knn_test4.score(X_test, y_test)
      5 print(f'{knn4_score:.2%}')

98.00%
```

Як бачимо по результатах, отримали приблизно однакові результати для даного набору даних. Але, я вважаю, що найбільш оптимальні значення гіперпараметра є значення 3-5. Ці значення будуть робити нашу модель стійкими до викидів та аномалій. Саме через це значення гіперпараметра за замовчуванням у бібліотеці scikit-learn є 5, бо для більшості вибірок вона є найбільш релевантним.

### **Висновок:**

У рамках лабораторної роботи було вивчено метод класифікації за допомогою k-найближчих сусідів (KNN) на прикладі набору даних Digits. Основною метою було підібрати оптимальні значення гіперпараметрів для побудови класифікатора.

Під час виконання лабораторної роботи були проведені експерименти з різними значеннями гіперпараметра `n_neighbors`, який визначає кількість найближчих сусідів для прийняття рішення. Варіювання цього гіперпараметра дозволило оцінити його вплив на продуктивність моделі класифікації та точність результатів.

З метою знаходження оптимального значення `n_neighbors`, було розглянуто декілька значень, таких як 1, 3, 5, 7 і 9. Це дозволило оцінити вплив різних розмірів сусідського околу на точність класифікації. В результаті, визначено оптимальне значення `n_neighbors=5`, яке максимізує точність класифікації на наборі даних Digits.