Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №2

з дисципліни «Вступ до машинного навчання»

Тема «Логістична регресія»

Варіант №17

Студента 4-го курсу НН ІАТЕ гр. ТР-12
Ковальова Олександра
Перевірив: вик. Ліскін В’ячеслав Олегович

КИЇВ 2025

Хід роботи.

1. **Відкрити та зчитати наданий файл з даними.**

Файл був прочитаний за допомогою функції read_csv. Додатково вказуємо сепаратор, тому що дані в файлі розділені через символ крапки з комою.

```
1  import pandas as pd
2  df = pd.read_csv("data/Input_Lab-2_3_4_WQ-R.csv", sep=";")
   ✓ [1] 299ms
```

2. **Визначити та вивести кількість записів, а також кількість полів та їх тип у завантаженому наборі даних.**

Кількість полів та записів можна визначити за допомогою поля shape, а типи за допомогою поля dtypes.

```
1  num_records = df.shape[0]
2  num_fields = df.shape[1]
3  print(f"Кількість записів (рядків): {num_records}")
4  print(f"Кількість полів (стовпців): {num_fields}")
5  print(df.dtypes)
   ✓ [2] < 10 ms
   Кількість записів (рядків): 1599
   Кількість полів (стовпців): 12
   fixed acidity           float64
   volatile acidity        float64
   citric acid             float64
   residual sugar          float64
   chlorides               float64
   free sulfur dioxide     float64
   total sulfur dioxide    float64
   density                 float64
   pH                      float64
   sulphates               float64
   alcohol                 float64
   quality                   int64
   dtype: object
```

3. **Вивести перші 10 записів набору даних.**
З цим питанням може допомогти функція head.

```
display(df.head(10))
```
✓ [3] 21ms

10 rows × 10 rows × 12 cols — Static Output

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dio
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	

4. **Додати до набору даних атрибут HighQuality, який дорівнює 1, якщо quality більше або дорівнює шести, та 0 в інших випадках.**

```
df["HighQuality"] = (df["quality"] >= 6).astype(int)
display(df.head(10))
```
✓ [4] 16ms

10 rows × 10 rows × 13 cols — Static Output

dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	HighQuality
11.0	34.0	0.9978	3.51	0.56	9.4	5	0
25.0	67.0	0.9968	3.20	0.68	9.8	5	0
15.0	54.0	0.9970	3.26	0.65	9.8	5	0

5. **Вивести перші 10 записів набору даних та видалити після цього атрибут quality з набору даних.**

```
df.drop(columns=["quality"], inplace=True)
display(df.head(10))
```
✓ [5] 17ms

10 rows × 10 rows × 12 cols — Static Output

free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	HighQuality
11.0	34.0	0.9978	3.51	0.56	9.4	0
25.0	67.0	0.9968	3.20	0.68	9.8	0

6. **Перемішати набір даних та розділити його на навчальну (тренувальну) та тестову вибірки, використовуючи функцію ShuffleSplit.**

```
from sklearn.model_selection import ShuffleSplit

# Перемішування та розбиття на тренувальну (80%) і тестову (20%) вибірки
splitter = ShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
train_idx, test_idx = next(splitter.split(df))

df_train = df.iloc[train_idx].reset_index(drop=True)
df_test = df.iloc[test_idx].reset_index(drop=True)

print(f"Розмір тренувальної вибірки: {df_train.shape[0]}")
print(f"Розмір тестової вибірки: {df_test.shape[0]}")
```
✓ [6] 748ms

```
Розмір тренувальної вибірки: 1279
Розмір тестової вибірки: 320
```

7. **Використовуючи відповідні функції бібліотеки scikit-learn, збудувати класифікаційну модель логістичної регресії та навчити її на тренувальній вибірці, вважаючи, що цільова характеристика визначається стовпчиком HighQuality, а всі інші виступають в ролі вихідних аргументів.**

Розділяємо ознаки та цільові змінні. X_train і X_test містять всі стовпці, крім HighQuality (вхідні ознаки). y_train і y_test містять стовпець HighQuality (мітки класів: 0 або 1). Це потрібно, бо HighQuality – це цільова змінна (що ми намагаємось передбачити), а всі інші стовпці – вхідні дані. Далі, масштабуємо ознаки, бо масштабування важливе для алгоритмів, які використовують градієнтний спуск (як логістична регресія), бо воно приводить всі ознаки до одного масштабу та покращує збіжність моделі (зменшує вплив великорозмірних чисел). Після цього створюємо та навчаємо модель логістичної регресії.

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

# Виділяємо цільову змінну та вхідні характеристики
X_train = df_train.drop(columns=["HighQuality"])
y_train = df_train["HighQuality"]
X_test = df_test.drop(columns=["HighQuality"])
y_test = df_test["HighQuality"]

# Нормалізація ознак
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Модель логістичної регресії
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train_scaled, y_train)

✓ [7] 71ms
```

8. **Обчислити класифікаційні метрики збудованої моделі для тренувальної та тестової вибірки. Представити результати роботи моделі на тестовій вибірці графічно.**

Функція classification_report() виводить основні метрики класифікації, Precision (Точність): наскільки передбачення позитивного класу є правильними, Recall (Повнота): наскільки добре модель знаходить усі об'єкти позитивного класу, F1-score: середнє між precision і recall, Support: кількість об'єктів у кожному класі. Модель показує схожу точність на тренувальній і тестовій вибірках (~74-75%), що означає відсутність переобучення. Трохи нижча точність для класу 0, можливо, через незбалансовані дані. Модель працює достатньо стабільно, але можна покращити точність шляхом вибору найважливіших атрибутів, зміни параметрів моделі або додавання інших алгоритмів.

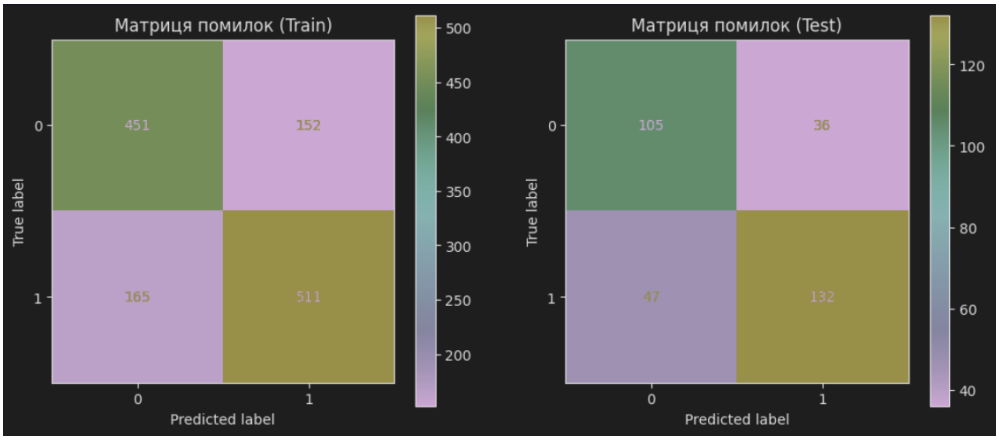
```python
from sklearn.metrics import classification_report, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Передбачення
y_train_pred = model.predict(X_train_scaled)
y_test_pred = model.predict(X_test_scaled)

# Вивід метрик
print("Класифікаційний звіт (тренувальна вибірка):")
print(classification_report(y_train, y_train_pred))

print("Класифікаційний звіт (тестова вибірка):")
print(classification_report(y_test, y_test_pred))

# Графічне представлення матриці помилок
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred, ax=ax[0])
ax[0].set_title("Матриця помилок (Train)")
ConfusionMatrixDisplay.from_predictions(y_test, y_test_pred, ax=ax[1])
ax[1].set_title("Матриця помилок (Test)")
plt.show()
✓ [8] 504ms
```

```
Класифікаційний звіт (тренувальна вибірка):
              precision    recall  f1-score   support

           0       0.73      0.75      0.74       603
           1       0.77      0.76      0.76       676

    accuracy                           0.75      1279
   macro avg       0.75      0.75      0.75      1279
weighted avg       0.75      0.75      0.75      1279

Класифікаційний звіт (тестова вибірка):
              precision    recall  f1-score   support

           0       0.69      0.74      0.72       141
           1       0.79      0.74      0.76       179

    accuracy                           0.74       320
   macro avg       0.74      0.74      0.74       320
weighted avg       0.74      0.74      0.74       320
```

Код також будує графічне представлення матриці помилок. Матриця помилок (ConfusionMatrixDisplay) покаже:

- True Positives (TP) – правильно передбачені "1".
- True Negatives (TN) – правильно передбачені "0".
- False Positives (FP) – помилкові передбачення "1", коли це "0".
- False Negatives (FN) – помилкові передбачення "0", коли це "1".

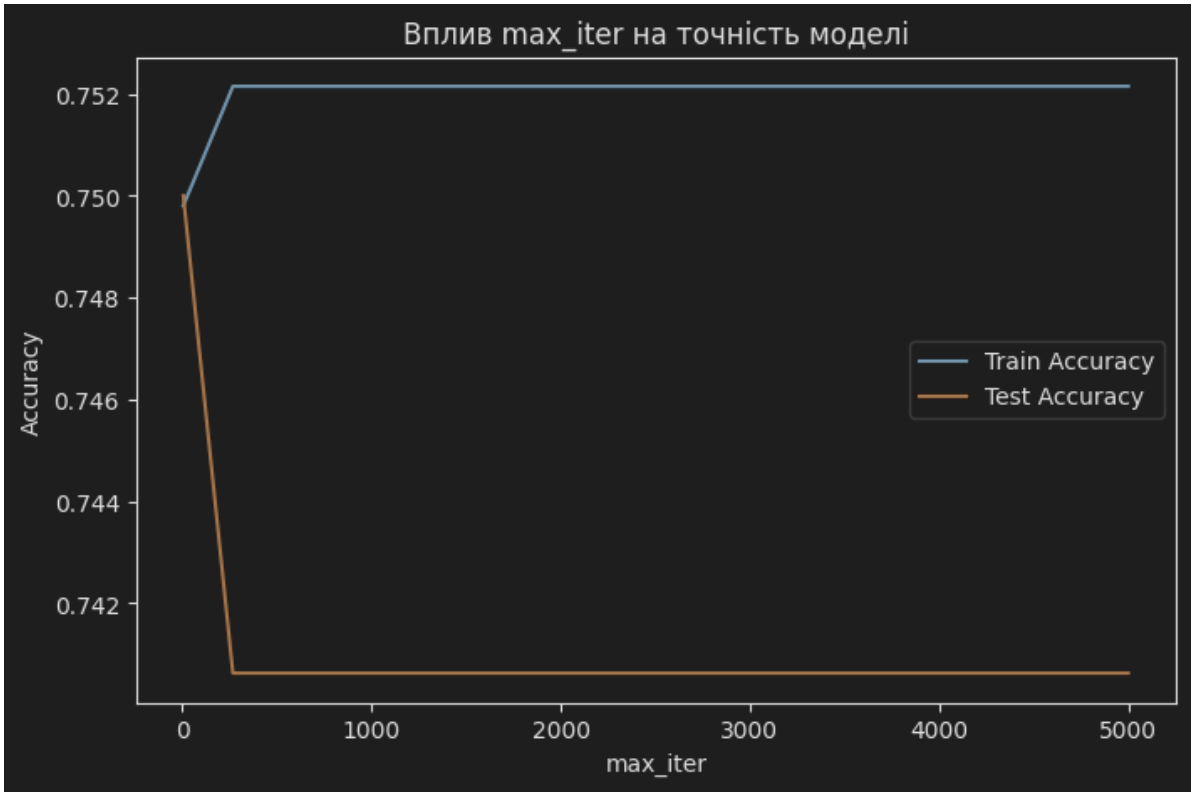
На графіку можна побачити, які помилки робить модель частіше, і, можливо, спробувати оптимізувати її. Це 2×2 матриця, оскільки наша модель вирішує задачу бінарної класифікації (два класи: 0 і 1). Матриця помилок допомагає зрозуміти, які саме помилки робить модель. У нашому випадку модель працює непогано, але помиляється у ~152 + 165 = 317 випадках. Можна спробувати покращити баланс між precision і recall або провести відбір найбільш значущих атрибутів для покращення роботи моделі.



9. З'ясувати вплив максимальної кількості ітерацій (від 5 до 5000) на результати класифікації. Результати представити графічно.

Графік точності (accuracy) залишається лінійним і незмінним після 150 ітерацій, це означає, що модель сходиться дуже швидко і подальше збільшення max_iter не впливає на результат.

```python
import numpy as np

iters = np.linspace(5, 5000, 20, dtype=int)
train_scores = []
test_scores = []

for it in iters:
    model = LogisticRegression(max_iter=it, random_state=42)
    model.fit(X_train_scaled, y_train)

    train_scores.append(model.score(X_train_scaled, y_train))
    test_scores.append(model.score(X_test_scaled, y_test))

plt.figure(figsize=(8, 5))
plt.plot(iters, train_scores, label="Train Accuracy")
plt.plot(iters, test_scores, label="Test Accuracy")
plt.xlabel("max_iter")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Вплив max_iter на точність моделі")
plt.show()

✓ [9] 341ms
```



10. Проаналізувати ступінь впливу атрибутів на результат класифікації. Збудувати класифікаційну модель логістичної регресії, залишивши від 3 до 5 найбільш важливих атрибутів та порівняти її результати із моделлю з п. 7.

Код визначає найбільш значущі атрибути (фічі) для моделі логістичної регресії, навчає нову модель лише на вибраних атрибутах і порівнює її результати з початковою моделлю. Відбувається визначення значущості атрибутів, відображення найбільш важливих, і потім навчання моделі на вибраних та її оцінка.

```python
import numpy as np

# Визначення важливості коефіцієнтів
feature_importance = np.abs(model.coef_).flatten()
sorted_idx = np.argsort(feature_importance)[-5:]  # Топ-5 ознак

top_features = X_train.columns[sorted_idx]
print("Найбільш значущі атрибути:", list(top_features))

# Навчання моделі тільки на вибраних атрибутах
X_train_selected = X_train_scaled[:, sorted_idx]
X_test_selected = X_test_scaled[:, sorted_idx]

model_selected = LogisticRegression(max_iter=1000, random_state=42)
model_selected.fit(X_train_selected, y_train)

# Порівняння якості моделі
print("Класифікаційний звіт для моделі з 5 атрибутами:")
print(classification_report(y_test, model_selected.predict(X_test_selected)))
✓ [10] 16ms
```

```
Найбільш значущі атрибути: ['free sulfur dioxide', 'sulphates', 'volatile acidity', 'total sulfur dioxide',
 'alcohol']
Класифікаційний звіт для моделі з 5 атрибутами:
              precision    recall  f1-score   support

           0       0.69      0.76      0.72       141
           1       0.79      0.73      0.76       179

    accuracy                           0.74       320
   macro avg       0.74      0.74      0.74       320
weighted avg       0.75      0.74      0.74       320
```

Найважливіші атрибути для класифікації якості вина:

- free sulfur dioxide (вільний діоксид сірки)
- sulphates (сульфати)
- volatile acidity (летюча кислотність)
- total sulfur dioxide (загальний діоксид сірки)
- alcohol (вміст алкоголю)

Модель, навчена лише на 5 атрибутах, дає таку ж точність (74%), як і модель з усіма ознаками.

Отриманий результат означає, що можна спростити модель, видаливши зайві атрибути без втрати точності. Точність скороченої моделі майже не відрізняється від повної моделі.

Точність (accuracy) залишається на рівні 74% у тестовій вибірці. Значення precision, recall і F1-score майже однакові. Recall класу 0 покращився (0.74 → 0.76), але Recall класу 1 трохи знизився (0.74 → 0.73). Це означає, що скорочена модель краще виявляє негативні зразки (клас 0), але трохи гірше визначає позитивні (клас 1). Precision залишився однаковим, що свідчить про незначну зміну в передбаченнях. Скорочена модель є кращою в плані інтерпретованості, оскільки використовує лише 5 атрибутів замість усіх. Це означає, що ми можемо досягти майже такої ж точності, використовуючи менше обчислювальних ресурсів і отримуючи простішу модель.

Чи варто використовувати скорочену модель? Якщо важлива простота та швидкість – так, оскільки точність не погіршилася. Якщо потрібно максимально використати всі доступні дані, можна залишити повну модель, але виграшу в точності немає.

Висновок: У ході виконання лабораторної роботи було проведено аналіз впливу кількості ітерацій на точність моделі логістичної регресії, а також досліджено важливість атрибутів для класифікації. Метою було побудувати класифікаційну модель для прогнозування якості вина на основі різних фізико-хімічних характеристик, використовуючи набір даних про вина.

Аналіз показав, що збільшення кількості ітерацій до певного порогу не призводить до значного покращення точності моделі, оскільки після 150 ітерацій точність стабілізувалась. Це свідчить про те, що для досягнення оптимальних результатів достатньо помірної кількості ітерацій, що дозволяє значно скоротити час навчання моделі.

Дослідження важливості атрибутів показало, що лише кілька характеристик, таких як концентрація сірчаної діоксиду, сульфатів і кислотності, мають найбільший вплив на результат класифікації. Використання тільки найбільш важливих атрибутів для побудови моделі не призвело до значного зниження точності, а навпаки, полегшило модель і зробило її більш інтерпретованою, що є важливим аспектом при роботі з реальними даними.

Загалом, виконана робота підтвердила ефективність застосування логістичної регресії для класифікації, а також продемонструвала, як можна оптимізувати модель шляхом вибору лише найбільш значущих атрибутів, зберігаючи високу точність і знижуючи складність моделі.