

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**З В І Т**

**Лабораторна робота №10  
з дисципліни  
«Комп’ютерні системи  
штучного інтелекту»**

Виконавець:

студент групи КІ-22м

Косей М.П.

Керівник:

викладач

Саяпін В.Г.

2023

## Лабораторна робота №10

**Тема:** Методи та засоби відображення розв'язків оптимізаційних задач.

**Мета:** Одержані знання роботи методів за допомогою яких можна автоматизовано приймати рішення.

### ХІД РОБОТИ

#### 1) Ознайомитись з теоретичними відомостями до лабораторної роботи

Метод повного перебору варіантів (**brute-force**) і дерева рішень (**Decision Trees**) є двома різними підходами до вирішення оптимізаційних задач.

У таблиці 1 дано порівняння цих двох методів.

**Таблиця 1.** Порівняння методу повного перебору варіантів і дерева рішень.

Аспекти	Дерево рішень	Метод повного перебору варіантів
<b>Складність обчислень</b>	Ефективний при великій кількості можливих варіантів	Експоненціальний ріст складності зі збільшенням розміру задачі
<b>Масштабованість</b>	Підходить для великих задач, структуроване зменшення простору пошуку	Непрактичний для великих задач, експоненціальний ріст кількості варіантів
<b>Інтерпретованість</b>	Інтерпретований, можливість візуалізації та розуміння	Немає структурованої інтерпретації
<b>Ефективність</b>	Ефективний для швидкого прийняття рішень, особливо для великих задач	Точний результат, але обчислювано витратний для великих задач

Дерева рішень використовується для прийняття рішень, що застосовуються в статистиці, аналізі даних та машинному навчанні.

Дерева рішень найбільш успішно застосовуються в наступних галузях:

- **Банківська справа:** оцінка кредитоспроможності клієнтів банку при наданні кредитів.
- **Промисловість:** контроль якості продукції (виявлення дефектів у готових товарах), безперебійні випробування (наприклад, перевірка якості зварки) та інші подібні завдання.
- **Медицина:** діагностика різних за складністю захворювань.
- **Молекулярна біологія:** аналіз структури амінокислот.
- **Торгівля:** класифікація клієнтів та товарів.

Цей інструмент допомагає вирішувати наступні задачі:

1. **Класифікація:** віднесення об'єктів до одного з наперед відомих класів. Цільова змінна повинна мати дискретні значення.
2. **Регресія:** прогнозування числового значення незалежної змінної для заданого вхідного вектора.
3. **Опис об'єктів:** набір правил у дереві рішень дозволяє компактно описувати об'єкти. Тому замість складних структур, що використовуються для опису об'єктів, можна зберігати дерева рішень.

Дерева рішень відносяться до типу навчання з вчителем.

У процесі навчання дерева рішень використовується набір навчальних даних, в якому кожному вхідному прикладу відповідає правильна вихідна мітка або класифікація. Модель дерева рішень навчається на цих парах вхід-вихід з метою прогнозування відповідей для нових вхідних даних.

Дерева рішень(**Decision Tree**) - це прості моделі, які мають гілки(**Branch**), вузли(**Node**) та листки(**Leaf Node**) і розбивають набір даних(**dataset**) на менші підмножини, що містять екземпляри з подібними значеннями. (див. рисунок 1).

Елементів дерева рішень:

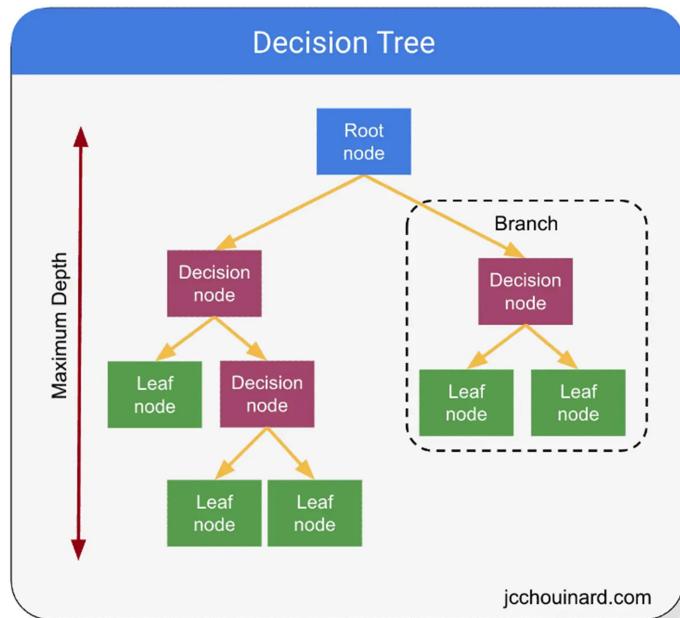
**Кореневий вузол (Root Node):** Перший вузол на шляху, з якого починаються всі рішення. Він не має батьківського вузла і має 2 дочірні вузли.

**Вузли рішень (Decision nodes):** Вузли, які мають 1 батьківський вузол і розгалужуються на дочірні вузли (рішення або листки).

**Листові вузли(Leaf nodes):** Вузли, які мають 1 батьківський вузол, але не поділяються далі (також відомі як термінальні вузли). Вони є вузлами, які дають прогноз або класифікацію.

**Гілки(Branches):** Частина дерева, також відома, як піддерево.

**Максимальна глибина(Maximum depth):** Максимальна глибина дерева визначається як найбільша кількість рівнів. Це найбільша кількість гілок або рішень, які можуть бути пройдені від кореневого вузла до будь-якого листового вузла в дереві.



**Рис. 1** Схема та визначення елементів дерева рішень.

Розберемо які властивості мають вузли дерев та алгоритм побудови дерева рішень на прикладі дерева для класифікації тварин на два класи: коти та собаки(див. рисунок 2, <https://towardsdatascience.com/decision-tree-an-algorithm-that-works-like-the-human-brain-8bc0652f1fc6>).

<b>WEIGHT (lbs)</b>	<b>HEIGHT (in)</b>	<b>LABEL</b>
8	8	
50	40	
8	9	
15	12	
9	9.8	

**Рис. 2 Навчальний датасет.**

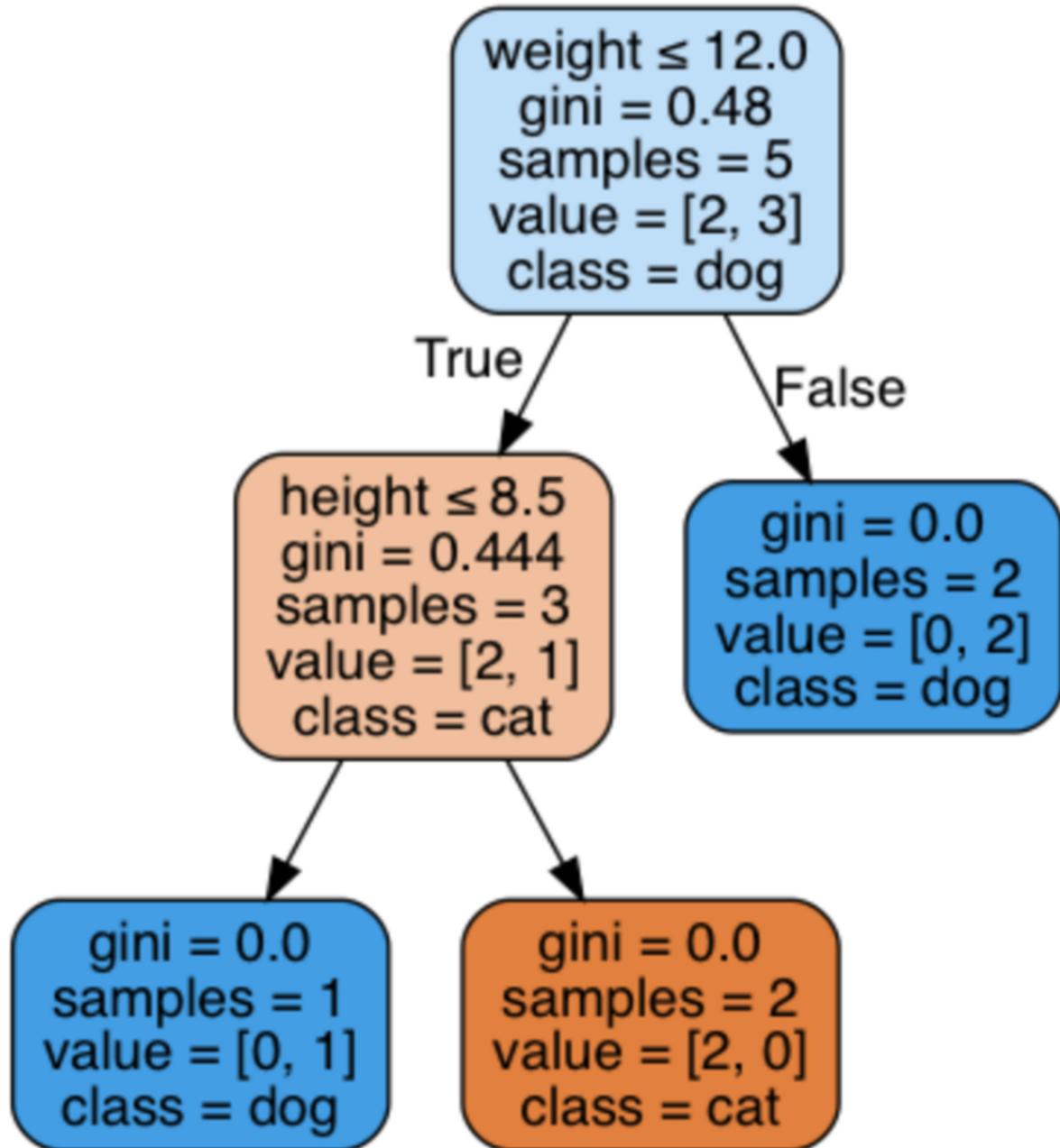
Вузол в дереві рішень має наступні властивості : (див. рисунок 3)

**Предикат:** Всі вузли, окрім листових вузлів мають асоційований предикат або умову, яка визначає, як розбити дані на підмножини. Предикат може бути умовою порівняння атрибуту з певним значенням або іншими логічними умовами (**weight <= 12, height <= 8.5**).

**Елементи датасету:** Вузол також містить підмножину даних, які відповідають певній умові, встановленій предикатом. Ці елементи датасету використовуються для подальшої рекурсивної побудови піддерев (**samples = 5, ...; values = [2,3] .....**).

Листовий вузол в дереві рішень має наступні властивості:

**Прогнозоване значення:** У випадку задачі класифікації це може бути конкретна мітка класу, до якого належать входні дані. У випадку задачі регресії це може бути числове значення, яке прогнозується моделлю (**class = dog, class = dog**).



**Рис. 3** [Приклад дерева для визначення класу тварин: кіт або собака.](#)

Алгоритм побудови дерева рішень може варіюватись в залежності від використовуваного методу (див. рисунки 4).

Один з найпоширеніших алгоритмів побудови дерева рішень - це алгоритм **ID3** (*Iterative Dichotomiser 3*) або його вдосконалена версія, така як **C4.5** або **CART** (*Classification and Regression Trees*).

Основні кроки алгоритму побудови дерева наступні:

- Вибір кореневого вузла:** Визначення, який атрибут (признак) найкраще розділяє навчальні дані на підмножини. Використовується принцип жадібної максимізації приросту інформації. На кожному кроці алгоритм вибирає атрибут, за яким відбувається розбиття, таким чином, щоб приріст інформації (зменшення ентропії) був максимальним (індекс **Gini** для задач класифікації). Далі процедура рекурсивно повторюється, поки ентропія не стане рівною нулю або якісь невеликій величині

(якщо дерево не підлаштовується ідеально під навчальну вибірку, щоб уникнути перенавчання).

**2. Розділення навчальних даних:** Розділення навчальних даних на підмножини на основі значення вибраного атрибута. Кожна підмножина відповідає одній гілці дерева.

**3. Рекурсивна побудова піддерев:** Повторення кроків 1 і 2 для кожної підмножини даних (гілки), створених на попередньому кроці. Цей процес продовжується досягнення заданої умови зупинки, такої як досягнення максимальної глибини, вичерпання атрибутів або інших критеріїв зупинки.

**4. Призначення значень листовим вузлам:** Визначення значень листових вузлів дерева, які відповідають прогнозованим класам або числовим значенням відповідно до задачі (класифікація або регресія).

Алгоритм побудови дерева може мати варіації та додаткові етапи в залежності від конкретної реалізації та використовуваних методів.

Принципово важливим є розділення даних на основі атрибутів та визначення оптимального розбиття для кожного вузла дерева.

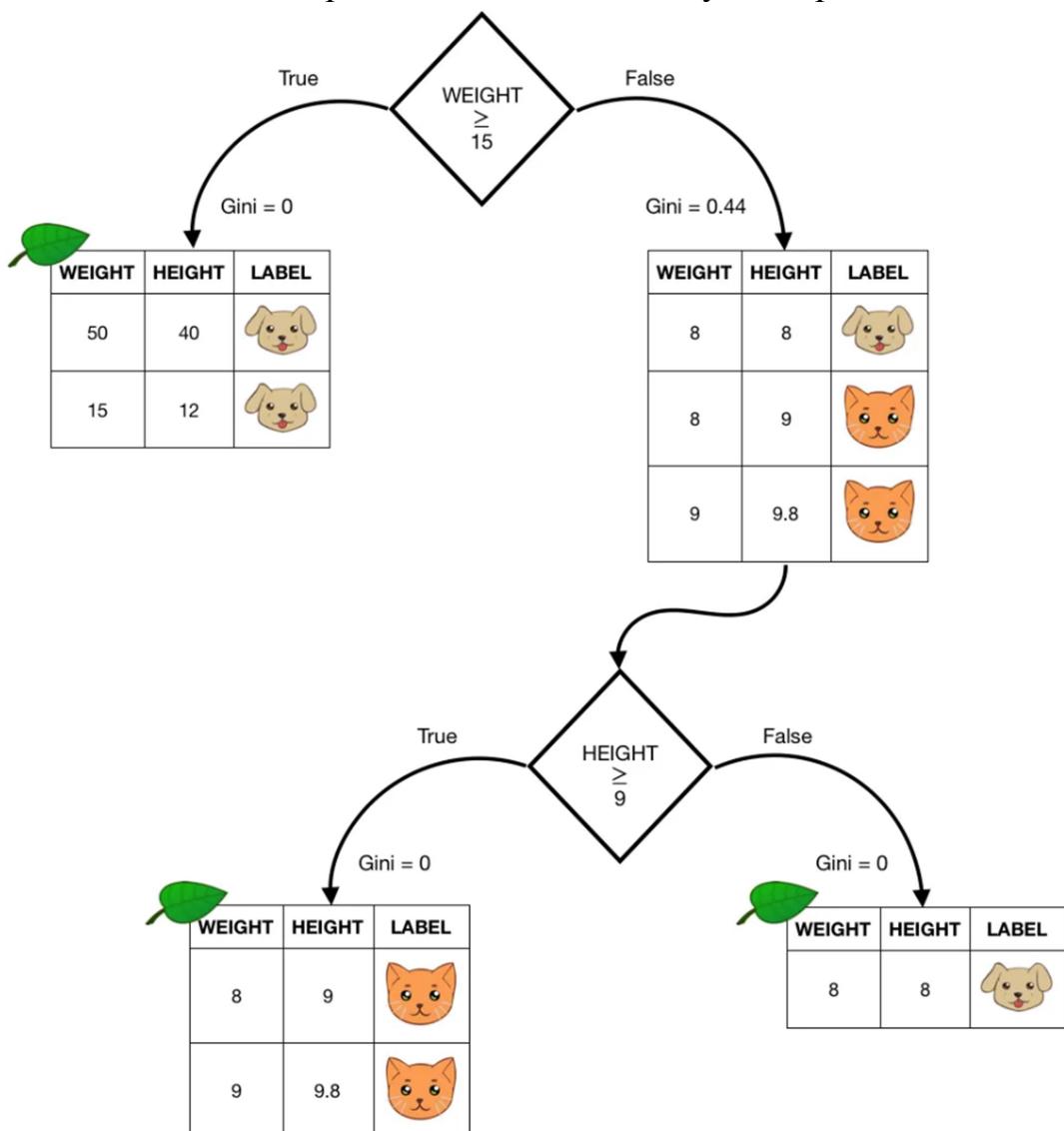


Рис. 4 [Побудова дерева для визначення класу тварин: кіт або собака.](#)

**2) Використовуючи будь-яку мову програмування розробити додаток який буде моделювати дерево перебирання для булевого виразу.**

В попередній лабораторній роботі № 8 за варіантом 8 необхідно було запрограмувати штучну нейрону мережу з двома виходами, щоб отримати значення:

- першої функції стрілка Пірса ( $\text{NOR} \downarrow$ ) на першому виході;
- другої функції штрих Шеффера ( $\text{NAND} \uparrow$ ) на другому виході.

Зробимо теж саме за допомогою дерева рішень.

Аналізуючи таблицю істинності функції (**таблиця 2**) і використовуючи алгоритм побудови дерев рішень, можна зводити задачу до задачі класифікації, де дерево рішень визначає правила класифікації об'єктів на основі їх характеристик **A, B**.

Кількість класів три: **11, 01, 00**.

**Таблиця 2.** Таблиця істинності для булевої логічної функції (навчальний датасет).

Вхідні змінні		Вихідні змінні	
A	B	$\text{NOR} \downarrow$	$\text{NAND} \uparrow$
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

Використовуємо мову програмування **Python** та проект [JupyterLab](#).



```

import pandas as pd
from sklearn.tree import DecisionTreeClassifier

# Створення датафрейму з вхідними даними
data = pd.DataFrame({
    'A': [0, 0, 1, 1],
    'B': [0, 1, 0, 1],
    'NOR': [1, 0, 0, 0],
    'NAND': [1, 1, 1, 0]
})

data.head()

```

	A	B	NOR	NAND
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1
3	1	1	0	0

```

# Створення стовпця "Class" на основі конкатенації стовпців NOR і NAND
data['Class'] = data['NOR'].astype(str) + data['NAND'].astype(str)

data.head()

```

	A	B	NOR	NAND	Class
0	0	0	1	1	11
1	0	1	0	1	01
2	1	0	0	1	01
3	1	1	0	0	00

```
# Розділення датафрейму на ознаки (X) та мітки класу (Y)
X = data[['A', 'B']]
Y = data['Class']
```

```
# Створення об'єкту дерева рішень
clf = DecisionTreeClassifier()

# Побудова дерева рішень на основі вхідних даних
clf.fit(X, Y)

# Класифікація нового об'єкту
new_object = data = pd.DataFrame({
    'A': [0, 0, 1, 1],
    'B': [0, 1, 0, 1],
}) # Новий об'єкт, для якого потрібно зробити прогноз

new_object.head()
```

	A	B
0	0	0
1	0	1
2	1	0
3	1	1

```
prediction = clf.predict(new_object)
print("Прогнозована мітка класу для нового об'єкту:", prediction)
```

Прогнозована мітка класу для нового об'єкту: ['11' '01' '01' '00']

```

from sklearn.tree import export_graphviz
import graphviz

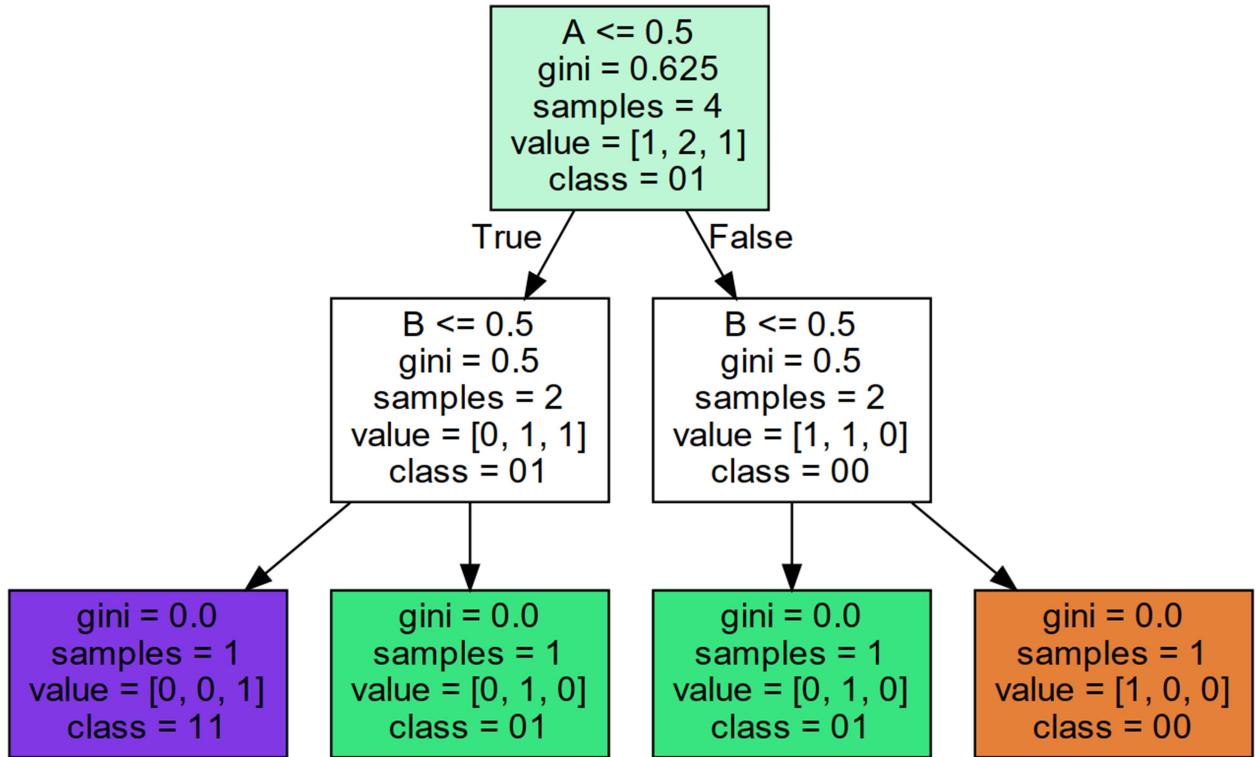
# Генерація коду для візуалізації дерева рішення
dot_data = export_graphviz(clf, out_file=None, feature_names=X.columns, class_names=clf.classes_, filled=True)

# Створення об'єкту графу на основі згенерованого коду
graph = graphviz.Source(dot_data)

# Відображення графічного представлення дерева
graph.render(filename='decision_tree', format='png', cleanup=True)
graph.view()

'decision_tree.pdf'

```



### 3) Розглянемо більш складний приклад та побудуємо дерево рішень.

Будемо працювати з набором даних для задачі класифікації - дані про серцево-судинні захворювання (<https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset?resource=download>).

У цій задачі пропонується передбачити наявність серцево-судинних захворювань на основі результатів класичного медичного огляду.

```
full_df = pd.read_csv('cardio.csv', sep=';')
full_df.head()
```

	<b>id</b>	<b>age</b>	<b>gender</b>	<b>height</b>	<b>weight</b>	<b>ap_hi</b>	<b>ap_lo</b>	<b>cholesterol</b>	<b>gluc</b>	<b>smoke</b>	<b>alco</b>	<b>active</b>	<b>cardio</b>
<b>0</b>	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
<b>1</b>	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
<b>2</b>	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
<b>3</b>	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
<b>4</b>	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

Датасет містить три групи ознак:

- Об'єктивні ознаки:
  - Вік (у днях);
  - Зріст;
  - Вага;
  - Стать;
- Результати вимірювання:
  - Верхній та нижній артеріальний тиск;
  - Рівень холестерину (три групи: норма, вище норми, значно вище норми);
  - Рівень глюкози (три групи: норма, вище норми, значно вище норми);
- Суб'єктивні ознаки (бінарні):
  - Куріння;
  - Споживання алкоголю;
  - Фізична активність.

## Робота з даними

Перевіримо чи необхідно очищення та обробка даних

```
# інформація про дані - назви стовпців, кількість, типи даних  
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 70000 entries, 0 to 69999  
Data columns (total 13 columns):  
 #   Column      Non-Null Count  Dtype     
 ---  --          --          --       --         
 0   id          70000 non-null   int64    
 1   age         70000 non-null   float64  
 2   gender      70000 non-null   int64    
 3   height      70000 non-null   int64    
 4   weight      70000 non-null   float64  
 5   ap_hi        70000 non-null   int64    
 6   ap_lo        70000 non-null   int64    
 7   cholesterol  70000 non-null   int64    
 8   gluc         70000 non-null   int64    
 9   smoke        70000 non-null   int64    
 10  alco         70000 non-null   int64    
 11  active        70000 non-null   int64    
 12  cardio       70000 non-null   int64    
 dtypes: float64(2), int64(11)  
memory usage: 6.9 MB
```

```
#Переведемо вік у роки  
full_df['age'] = round(full_df['age'] / 365)  
full_df.age = full_df.age.astype(np.int64)  
full_df.head()
```

	<b>id</b>	<b>age</b>	<b>gender</b>	<b>height</b>	<b>weight</b>	<b>ap_hi</b>	<b>ap_lo</b>	<b>cholesterol</b>	<b>gluc</b>	<b>smoke</b>	<b>alco</b>	<b>active</b>	<b>cardio</b>	
<b>0</b>	0	50	2	168	62.0	110	80		1	1	0	0	1	0
<b>1</b>	1	55	1	156	85.0	140	90		3	1	0	0	1	1
<b>2</b>	2	52	1	165	64.0	130	70		3	1	0	0	0	1
<b>3</b>	3	48	2	169	82.0	150	100		1	1	0	0	1	1
<b>4</b>	4	48		156	56.0	100	60		1	1	0	0	0	0

```
#наменімо представлення ваги з float64 на цілі числа.  
full_df.weight = full_df.weight.astype(np.int64)  
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 70000 entries, 0 to 69999  
Data columns (total 13 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          --          --  
 0   id          70000 non-null   int64    
 1   age         70000 non-null   int64    
 2   gender      70000 non-null   int64    
 3   height      70000 non-null   int64    
 4   weight      70000 non-null   int64    
 5   ap_hi       70000 non-null   int64    
 6   ap_lo       70000 non-null   int64    
 7   cholesterol 70000 non-null   int64    
 8   gluc        70000 non-null   int64    
 9   smoke       70000 non-null   int64    
 10  alco        70000 non-null   int64    
 11  active      70000 non-null   int64    
 12  cardio      70000 non-null   int64    
 dtypes: int64(13)  
 memory usage: 6.9 MB
```

```
full_df.weight.describe()
```

```
count    70000.000000  
mean     74.204329  
std      14.395953  
min      10.000000  
25%     65.000000  
50%     72.000000  
75%     82.000000  
max      200.000000  
Name: weight, dtype: float64
```

```
#Найхудіша доросла людина на Землі важить приблизно 23 кг, тому всі інші дані можуть бути викидами  
full_df = full_df[full_df.weight > 30]  
full_df.info()
```

```
full_df.weight.describe()
```

```
count      69990.000000
mean       74.211587
std        14.383905
min        31.000000
25%        65.000000
50%        72.000000
75%        82.000000
max        200.000000
Name: weight, dtype: float64
```

```
#нормальний діапазон систолічного артеріального тиску може бути від 70 до 200
```

```
full_df = full_df[(full_df.ap_hi < 200) & (full_df.ap_hi > 70 )]
full_df.ap_hi.describe()
```

```
count      69572.000000
mean       126.823420
std        16.626402
min        80.000000
25%        120.000000
50%        120.000000
75%        140.000000
max        199.000000
Name: ap_hi, dtype: float64
```

```
full_df.ap_lo.describe()
```

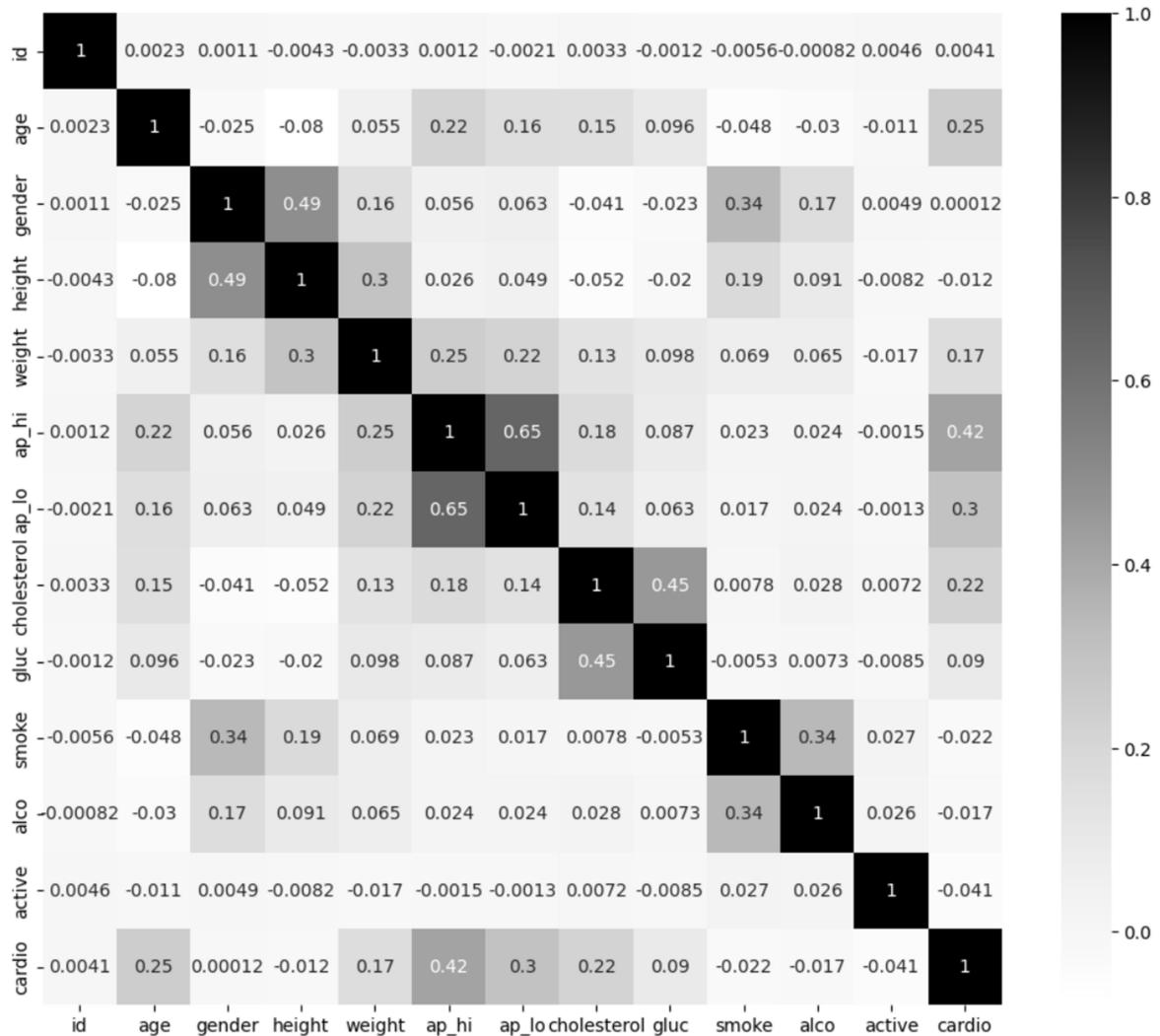
```
count      69572.000000
mean        96.114543
std         183.297261
min         0.000000
25%        80.000000
50%        80.000000
75%        90.000000
max       10000.000000
Name: ap_lo, dtype: float64
```

```
#діастолічний тиск значно перевищує середній діапазон 40:100
full_df = full_df[(full_df.ap_lo < 100) & (full_df.ap_lo > 40)]
full_df.ap_lo.describe()
```

```
count      63883.000000
mean        79.776764
std         7.661501
min         45.000000
25%        80.000000
50%        80.000000
75%        80.000000
max       99.000000
Name: ap_lo, dtype: float64
```

Подивимось кореляцію між ознаками

```
import seaborn as sns
plt.figure(figsize = (12,10))
sns.heatmap(full_df.corr() , cmap = 'Greys' , annot = True)
```



```

y = full_df['cardio'].astype(str)
X = full_df.drop(['cardio'] , axis =1)

#Розділяємо датасет на тренувальний та тестовий
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

```

```

from sklearn import tree

dec = tree.DecisionTreeClassifier(max_depth=3)

dec.fit(X_train, y_train)
scores = dec.score(X_test, y_test)
print("Accuracy score:", scores)

```

Accuracy score: 0.716630300730481

```
num_nodes = dec.tree_.node_count
depth = dec.get_depth()
num_leaves = dec.get_n_leaves()

print("Кількість вузлів: ", num_nodes)
print("Глибина дерева: ", depth)
print("Кількість листків: ", num_leaves)
```

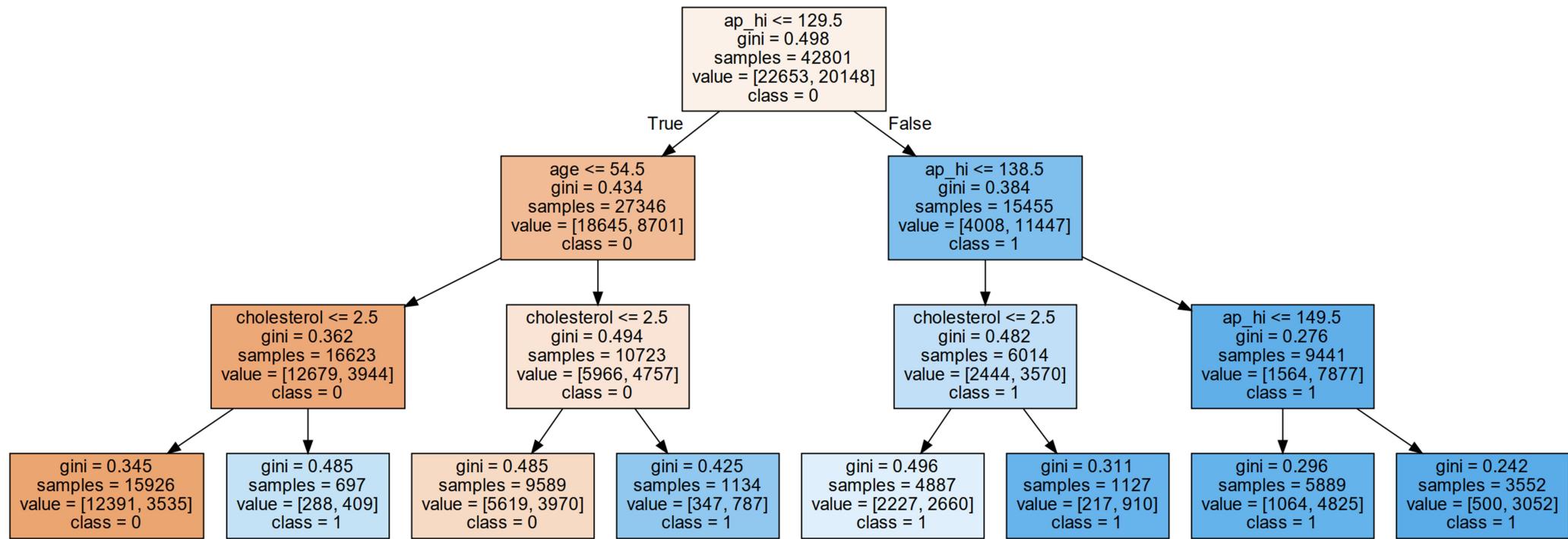
```
Кількість вузлів: 15
Глибина дерева: 3
Кількість листків: 8
```

```
# Візуалізація дерева рішень
dot_data = tree.export_graphviz(dec, out_file=None,
                                feature_names=X.columns,
                                class_names=dec.classes_,
                                filled=True)

# Створення об'єкту графу на основі згенерованого коду
graph = graphviz.Source(dot_data)

# Відображення графічного представлення дерева
graph.render(filename='decision_tree_cardio', format='png', cleanup=True)
graph.view()

'decision_tree_cardio.pdf'
```



## **ВИСНОВКИ**

**В результаті виконаної лабораторної роботи одержати знання для побудови дерев рішень за допомогою яких можна автоматизовано приймати рішення.**

**Усі матеріали викладені у репозіторії GitHub, за посиланням**  
<https://github.com/Max11mus/Artifition-Intelect-Lab10.git>.