# ЛАБОРАТОРНА РОБОТА № 4
# ЗНАЙОМСТВО З ПРОГРАМОЮ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ WEKA ТА ПІДГОТОВКА ДАНИХ

## Мета роботи

Ознайомитися та отримати навички роботи з бібліотекою data mining алгоритмів WEKA. На практиці вивчити методи попередньої обробки даних для задач інтелектуального аналізу даних.

## Основні теоретичні відомості

WEKA (Waikato Environment for Knowledge Analysis) – бібліотека алгоритмів машинного навчання для вирішення завдань інтелектуального аналізу даних (data mining). Система дозволяє безпосередньо застосовувати алгоритми до вибірок даних, а також викликати алгоритми з програм на мові Java.

WEKA – продукт університету Уайкато (Нова Зеландія), який вперше був випущений в його сучасному вигляді в 1997 році. WEKA поширюється по ліцензії GNU General Public License (GPL). Це програмне забезпечення написано на мові Java та забезпечує графічний користувальницький інтерфейс для роботи з файлами даних і генерації візуальних результатів (у вигляді таблиць і графіків). Крім того є можливість інтегрувати WEKA, як і будь-яку іншу бібліотеку, у свої власні додатки, наприклад, для автоматизації аналізу даних на стороні сервера, використовуючи стандартний API.

Цілі проекту – створити сучасне середовище для розробки методів машинного навчання та застосування їх до реальних даних, зробити методи машинного навчання доступними для повсюдного застосування. Передбачається, що за допомогою даного середовища фахівець у прикладній області зможе використовувати методи машинного навчання для вилучення корисних знань безпосередньо з даних дуже великого обсягу.

Користувачами WEKA є дослідники в області машинного навчання і прикладних наук. Вона також широко використовується в навчальних цілях.

Основні можливості GUI інтерфейсу програми WEKA наведено в додатку А.

Програма дозволяє завантажити та провести попередню обробку даних (*Preprocess*), вирішити задачу класифікації або регресії (*Classify*),

кластеризації **(***Cluster***),** пошуку асоціативних правил **(***Associate***),** відбору атрибутів **(***Select Attributes***)** та візуалізації **(***Visualize***).**

Дані для аналізу в WEKA можуть бути завантажені з файлу, із віддаленого джерела, з бази даних або дані можна згенерувати за допомогою математичної моделі.

***Формат файлів даних ARFF***

Основний формат файлів даних, який використовується в WEKA, – це ARFF. У каталозі data, який знаходиться в каталозі встановленої програми, можна знайти приклади arff-файлів.

ARFF файл є ASCII текстовим файлом, який описує список об'єктів із загальними ознаками (атрибутами). Структурно такий файл розділяється на дві частини: заголовок і дані.

У заголовку описується ім'я даних та їх метадані (імена атрибутів і їх типи). Наприклад,

```
% коментар
@RELATION myproblem
@ATTRIBUTE firstfeature REAL
@ATTRIBUTE class {A,B}
```

У другій частині представлені самі дані. Наприклад,

```
@ DATA
1.1,A
```

Заголовок містить інформацію про ім'я файлу і метадані про представлені у ньому даних. Ім'я описується в наступному форматі:

```
@relation <ім’я>
```

Іменем може бути будь-яка послідовність символів. Якщо ім'я містить пробіли, то воно має бути взято в лапки. Наприклад,

```
@relation weather
@relation ‘weather nominal’
```

Метадані описують атрибути представлених у файлі даних. Інформація про кожний атрибут записується в окремому рядку і включає ім'я атрибуту і його тип. Очевидно, що всі імена повинні бути унікальними. Порядок їх опису повинен збігатися з порядком колонок в описі самих даних. Загальний формат опису атрибуту наступний:

```
@attribute <ім’я атрибута> <тип атрибута>
```

Наприклад,

```
@attribute temperature real
```

Ім'я атрибуту має починатися з символу @. У разі якщо в імені містяться пробіли, воно має бути взято в лапки.

Поле <тип> може мати одне з таких значень:

– real;
– integer;
– <категорія>;
– string;
– date [<формат дати>].

Типи real і integer є числовими. Категоріальні типи описуються переліком категорій (можливих значень). Наприклад:

```
@attribute outlook {sunny, overcast, rainy}
```

Дані представляються в ARFF форматі у вигляді списку значень атрибутів об'єктів після тегу @ data. Кожен рядок списку відповідає одному об'єкту, кожна колонка – атрибуту, описаному в заголовку. Часто в термінології data mining такі рядки називають векторами.

Дані можуть містити припущення (невідомі) значення. У ARFF вони представляються символом «?», Наприклад:

```
@data
4.4,?,1.5,?,Iris-setosa
```

Строкові дані, у разі якщо вони містити символи, що розділяють, повинні братися в лапки. Наприклад,

```
@relation LCCvsLCSH
@attribute LCC string
@attribute LCSH string

@data
AS262, 'Science - Soviet Union - History.'
```

При описі дати можна вказати формат, в якому вона записується. Дати також повинні братися в лапки.

```
@relation Timestamps
@attribute timestamp DATE "yyyy-MM-dd HH:mm:ss"
@data
"2001-04-03 12:12:12"
```

Кожен набір даних, який використовується в лабораторних роботах, представлений у форматі ARFF. На початку кожного файлу міститься вичерпна інформація про задачу, представлену цим набором даних.

***Попередня обробка даних***

Дані володіють таким параметром як якість, яке включає наступні параметри: точність, повнота, несуперечність, своєчасність, достовірність та інтерпретованість.

Для підвищення якості даних і підготовки їх до обробки

методами інтелектуального аналізу існує кілька технологій попередньої обробки даних.

До основних задач попередньої обробки даних відносяться наступні.

*Задача очищення даних*, яка використовується для заповнення пропущених значень, видалення шумів, видалення суперечливості, ідентифікації та видалення викидів.

*Задача інтеграції даних із різних джерел* (баз даних, кубів даних, файлів) в одне узгоджене сховище. Ця задача передбачає об'єднання даних і усунення неузгодженостей, дублікатів, конфліктів.

*Задача проріджування та стиснення даних* використовується для зменшення розміру даних з мінімізацією втрати інформації. Ця задача включає зниження розмірності даних (відбір атрибутів) і чисельне зменшення (побудова мат. моделей для значень атрибутів).

*Задача перетворення даних*. До неї відносяться нормалізація, дискретизація, квантування, згладжування, агрегація даних, відображення даних за допомогою ядерних функцій.

Також до попередньої обробки даних можна віднести *перетворення задачі множинної класифікації в бінарну*.

***Відбір атрибутів***

У більшості практичних ситуацій набори даних містять занадто багато атрибутів, що збільшує час навчання алгоритмів. При цьому деякі з атрибутів є незначущими чи надмірними. Таким чином дані повинні бути попередньо оброблені з метою відбору деякої мінімальної підмножини атрибутів для навчання.

Для вибору хорошої підмножини атрибутів існує два підходи.

Перший з них заснований на незалежній оцінці статистичних чи якихось інших характеристиках набору даних. Він називається фільтрацією і відбувається до початку безпосереднього аналізу даних.

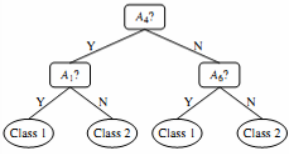
У другому підході відбір підмножини атрибутів виконується всередині методів інтелектуального аналізу. Такий підхід називається методом обгортки (wrapper method), тобто алгоритм навчання «обгорнутий» в процедуру відбору атрибутів.

Самі методи інтелектуального аналізу також можуть бути використані для відбору атрибутів. Наприклад, можна застосувати алгоритм побудови дерев рішень до повного набору даних і потім залишити в наборі тільки ті атрибути, які використані в побудованому дереві. Слід зауважити, що даний відбір атрибутів не дасть ніякого

ефекту при побудові нового дерева, проте виявиться корисним при використанні інших методів аналізу. Інша можливість - це застосувати до даних алгоритм, який будує лінійну модель (наприклад, метод опорних векторів), і ранжувати атрибути на підставі величин коефіцієнтів моделі. Атрибути з найменшими коефіцієнтами можуть бути відкинуті. Дану процедуру можна повторити кілька разів. Крім того для відбору атрибутів можуть бути застосовані методи аналізу, засновані на порівнянні близькості екземплярів вибірки. Для порівняння беруться сусідні примірники однакових і різних класів. Якщо у примірників одного класу значення певної атрибути різні, то можна припустити, що даний атрибут є незначущою і її вага повинна бути зменшена. З іншого боку, якщо у екземплярів різних класів атрибут має різні значення, то даний атрибут значимий і його вага повинна бути збільшена. Після повтору даної процедури кілька разів, відбувається відбір атрибутів з найбільшими вагами. До недоліків даного методу можна віднести той факт, що даний метод не зможе визначити надлишкові атрибути, пов'язані тісним кореляційним зв'язком.

Зазвичай пошук в просторі атрибутів відбувається в одному з двох напрямків: зверху вниз (починаючи з повного набору атрибутів і відкидаючи на кожному кроці найгірший з них) або знизу вгору (починаючи з порожньої множини атрибутів і додаючи найкращий з решти) (табл. 1.1).

Таблиця 1.1 – Пошук в просторі атрибутів

| **Прямий вибір (forward selection)** | **Зворотне виключення (backward elimination)** | **Застосування дерев рішень** |
|---|---|---|
| Початкова множина атрибутів {A1, A2, A3, A4, A5} | | |
| {}<br>=> {A1}<br>=> {A1, A4}<br>=> {A1, A4, A5} | {A1, A2, A3, A4, A5}<br>=>{A1, A2, A4, A5}<br>=>{A1, A4, A5} |  |
| | | =>{A1, A4, A5} |

У деяких випадках для поліпшення точності класифікації та кращого розуміння атрибутів для вирішення поставленого завдання можлива побудова нового атрибуту на основі існуючих. Наприклад,

можна ввести новий атрибут «Площа» на основі існуючих атрибутів «висота» і «ширина».

***Пропущені значення***

При роботі з даними, в яких є пропущені значення атрибутів для деяких екземплярів, існують наступні стратегії поведінки.

1. Відкинути екземпляри з пропущеними значеннями. Такий підхід застосовується насамперед для даних, у яких відсутнє значення цільового атрибута (для задач класифікації).
2. Заповнити пропущені значення вручну.
3. Застосувати глобальну константу (наприклад, «Unknown»).
4. Використати деяке статистично розраховане по всій вибірці значення (середнє арифметичне, медіану, моду).
5. Використати статистичне значення, розраховане для примірників, що відносяться до того ж класу, як і розглянутий екземпляр.
6. Використати найбільш ймовірне значення для атрибуту. Це значення може бути розраховане за допомогою регресії, дерева рішень або інших математичних підходів.

***Нормалізація даних***

Одиниці виміру, що використовуються в деякому атрибуті, можуть вплинути на результати аналізу. Так, наприклад, перетворення одиниць вимірювання з метрів в дюйми для атрибуту «висота» або перетворення з кілограмів у фунти для атрибуту «вага» можуть призвести до різних результатів. У загальному випадку, вираз деякої атрибуту в дрібніших одиницях виміру приведе до більш широкого діапазону значень для цього атрибуту, що може призвести до більшої значущості або ж ваги даного атрибуту.

Щоб уникнути залежності від вибору одиниць вимірювання та надати всім атрибутам однакову вагу дані повинні бути нормалізовані або нормовані. Нормалізація передбачає перетворення даних таким чином, щоб діапазон значень, прийнятих атрибутом, зменшився або став рівним [-1; 1] або [0; 1]. Нормалізація найбільш корисна в задачах з застосуванням нейронних мереж та задачах, алгоритми яких засновані на обчисленні відстаней.

Існує багато методів нормалізації даних. Розглянемо деякі з них.

Нехай у нас є числовий атрибут A з виміряними значеннями $a_1$, $a_2$, …, $a_n$.

*Мінімаксна нормалізація*. Нехай $\min_a$ – мінімальне значення атрибуту, $\max_a$ – максимальне, [new_$\min_a$; new_$\max_a$] - новий діапазон для атрибуту, тоді:

$$a_i' = \frac{a_i - \min_a}{\max_a - \min_a}(new\_\max_a - new\_\min_a) + new\_\min_a.$$

*Нормалізація з нульовим середнім*. Значення атрибуту нормалізуються за допомогою математичного очікування $\bar{a}$ та стандартного відхилення $\sigma_a$ атрибуту:

$$a_i' = \frac{a_i - \bar{a}}{\sigma_a}.$$

Існує варіація нормалізації з нульовим середнім, в якому замість стандартного відхилення атрибуту використовує середнє абсолютне значення атрибуту:

$$S_a = \frac{1}{n}\left(|a_1 - \bar{a}| + |a_2 - \bar{a}| + ... + |a_n - \bar{a}|\right).$$

*Нормалізація за допомогою десяткової шкали*.

$$a_i' = \frac{a_i}{10^j},$$

де j - найменше ціле число, таке що ($|a_i|$)<1.

***Дискретизація числових атрибутів***

Дискретизація числових атрибутів є обов'язковою і необхідною у разі застосування алгоритмів інтелектуального аналізу, що працюють тільки з категоріальними атрибутами. Крім того, алгоритми, що працюють з числовими атрибутами часто дають кращі результати або ж працюють швидше, якщо значення атрибутів попередньо приведені до дискретної форми.

Методи дискретизації можуть бути класифіковані за двома параметрами:

– чи використовується в них інформація про класи: дискретизація з учителем (supervised discretization) або дискретизація без вчителя (unsupervised discretization);

- в якому напрямку відбувається дискретизація:
  - зверху-вниз (дискретизація починається з однієї або декількох точок поділу, а далі отримані інтервали рекурсивно розбиваються; метод розбиття);
  - знизу-вгору (спочатку всі значення атрибуту розглядаються як потенційні точки поділу, а далі сусідні значення рекурсивно об'єднуються, утворюючи інтервали; об'єднання).

***Вибірка (sampling)***

Вибірка застосовується в якості методу зменшення початкового набору даних з метою представлення великої вихідної множини екземплярів вибірки набагато меншою за розміром підмножиною.

Припустимо, що вихідний набір даних D містить N примірників. Розглянемо найбільш загальні шляхи зменшення його розміру.

*Проста випадкова вибірка без повернення*: з вихідного набору D випадковим чином вибирається S примірників (S<N), при цьому ймовірність вибору кожного примірника рівноймовірна.

*Проста випадкова вибірка з поверненням*: схожа на попередню, однак з тією відмінністю, що після вибору примірника, він повертається у вихідну вибірку і згодом знову може бути вибраний.

*Кластерна вибірка*: якщо вихідна вибірка згрупована в деякі роз'єднаним «кластери» (наприклад, сторінки з бази даних або дані з різних географічних джерел), то до кожного з таких кластерів може бути застосована проста випадкова вибірка.

*Стратифікована вибірка*: якщо вихідна вибірка несиметрична щодо розподілу класів і може бути розділена на страти, то проста випадкова вибірка застосовується до кожної страти окремо (наприклад, якщо дані представляють відомості про покупців різних вікових груп і при цьому кількість представників різних груп не однакова, такий підхід дозволить не втратити відомості про рідкісні групи покупців).

**Завдання на лабораторну роботу**

1. Побудуйте потік даних в модулі Knowledge flow як описано в додатку А.
2. В додатку Б оберіть вибірку для аналізу.
3. Використовуючи вкладки попередньої обробки даних та візуалізації, проведіть детальний опис вибірки даних. Вкажіть:
   - яке практичне завдання вирішується;

- скільки примірників у вибірці;
- атрибути, які характеризують екземпляри вибірки, їхні типи та опис;
- чи є екземпляри з відсутніми значеннями, чи є викиди в даних;
- який атрибут є цільовим, які значення він приймає, скільки екземплярів кожного класу в вибірці;
- подайте дані в графічному вигляді;
- наведіть перші 5 екземпляри вибірки.

4. Встановіть додатковий пакет scatterPlot3D та візуалізуйте дані.

**Контрольні питання**

1. Що таке інтелектуальний аналіз даних?
2. Що таке розвідувальний аналіз?
3. Для чого використовується програма WEKA, які її можливості.
4. Яке призначення модулів Explorer, Knowledge Flow, Experimenter, Command-Line Interface?
5. Опишіть формат arff файлу.
6. Опишіть призначення вкладок в модулі Explorer: Preprocess panel, Classify, Cluster, Associate, Select Attributes, Visualize.
7. Що таке генеральна сукупність і вибірка? Якими властивостями повинні володіти дані? Що таке репрезентативна вибірка?
8. Що розуміють під фільтрацією в Weka? В чому різниця між фільтрами атрибутів та фільтрами екземплярів? В чому різниця між unsupervised та supervised фільтрами?
9. Що таке якість даних? Яка мета підготовки даних до аналізу? Які завдання входять в підготовку даних?
10. Який атрибут даних називають цільовим?
11. Що таке значимий та незначимий атрибут? Що таке відбір атрибутів?
12. За допомогою яких фільтрів можна виконати наступні завдання підготовки даних:
    - перетворити тип атрибута;
    - нормалізувати значення числового атрибута;
    - найти та замінити відсутні значення в даних;
    - видалити всі екземпляри даних з заданим значенням

атрибута;
- створити новий атрибут;
- виконати відбір атрибутів;
- знайти викиди в даних;
- створити підвибірку даних.

**Зміст звіту**

1. Тема і мета роботи.
2. Завдання до роботи.
3. Результати виконання завдань.
4. Відповіді на контрольні запитання.
5. Висновки, що відображують результати виконання роботи та їх критичний аналіз.

## Додаток А
## Інтерфейс програми WEKA

Розглянемо можливості GUI інтерфейсу програми WEKA.

### Головне вікно програми

Основне вікно програми – це Weka GUI Chooser (рис. А.1). Опишемо більш докладно призначення керуючих елементів даного вікна.


Рисунок А.1 – Головне вікно програми WEKA

Основне вікно програми надає доступ до чотирьох модулів програми:

– **Explorer** – середовище для дослідження даних;

– **Experimenter** – середовище для проведення порівняльного аналізу роботи різних алгоритмів при обробці наборів даних;

– **KnowledgeFlow** – середовище, що підтримує таку ж функціональність, як і Explorer, але із застосуванням графічного представлення потоків даних;

– **Workbench** – середовище, яке об'єднує всі три наведені вище середовища в одне з можливістю перемикання подання;

– **SimpleCLI** – командний інтерфейс для безпосереднього виконання команд WEKA.

Головне меню програми складається з чотирьох пунктів.

**1. Program:**

– *LogWindow* – відкриває вікно логів, яке зберігає всю інформацію, виведену в потоки stdout або stderr;

– *Memory usage* – використання пам'яті;

– *Settings* – налаштування інтерфейсу;

– *Exit* – вихід.

**2. Tools:**

– *Package manager* – встановлення додаткових пакетів, що містять додаткові алгоритми, засоби візуалізації тощо;

– *ArffViewer* – редактор arff-файлів;

– SqlViewer – модуль перегляду баз даних;

– *Bayes net editor* – модуль для редагування, візуалізації та навчання Байесови мереж (Bayes nets).

**3. Visualization** – засоби візуалізації даних WEKA.

– *Plot* – відображення 2D-графіка набору даних;

– *ROC* – відображає раніше збережену ROC-криву;

– *TreeVisualizer* – відображає спрямовані графи, тобто дерева рішень;

– *GraphVisualizer* – візуалізує графіку для Байесових мереж;

– *BoundaryVisualizer* – дозволяє візуалізувати границі рішень класифікаторів у двох вимірах.

**4. Help** – розділ довідкової інформації.

– *Weka homepage* – домашня сторінка проекту WEKA;

– *HOWTOs, code snippets, etc*. – приклади, що стосуються розробки та використання WEKA;

– *Weka on Sourceforge* – сторінка проекту WEKA на Sourceforge.net;

– *SystemInfo* – значення деяких змінних середовища Java/WEKA.

## Модуль Explorer

Це основний модуль програми, який дозволяє завантажити і попередньо обробити дані (вкладка *Preprocess*), вирішити задачу класифікації або регресії (*Classify*), кластеризації (*Cluster*), пошуку асоціативних правил (*Associate*), відбору атрибутів (*Select Attributes*) і візуалізації (*Visualize*).

Кожна задача має свою вкладку в загальному вікні. Спочатку доступна тільки вкладка *Preprocess*, оскільки для виконання інших завдань потрібні дані. Зазначимо, що послідовність вкладок не завжди відповідає етапам вирішення задачі. Наприклад, після завантаження даних можна перейти до відбору атрибутів.

Внизу кожної вкладки відображається рядок статусу. Клік правою кнопкою миші на рядку статусу видає контекстне меню:

– *Memory information* – кількість доступної пам'яті;

– *Run garbage collector* – запуск збирача сміття Java, що очищає області пам'яті, які більше не використовуються, дозволяючи звільнити пам'ять для нових завдань.

Кнопка «*LOG*» дозволяє побачити лог подій що відбулися за час роботи WEKA.

У правій частині статусного рядка зображена пташка Уека. Якщо вона рухається, то програма робить обчислення, якщо сидить нерухомо, то програма знаходиться в режимі очікування. Після значка «X» відображається кількість запущених процесів.

### Завантаження і попередня обробка даних (Preprocess)

Спочатку на першій вкладці «*Preprocess*» вгорі вікна активні чотири кнопки, які дозволяють завантажити дані з файлу **(***Open file***),** з віддаленого джерела **(***Open URL***),** з бази даних **(***Open DB***)** або згенерувати модельні дані **(***Generate***)**.

Після завантаження файлу на панелі попередньої обробки даних з'являється інформація про дані (рис. А.2).

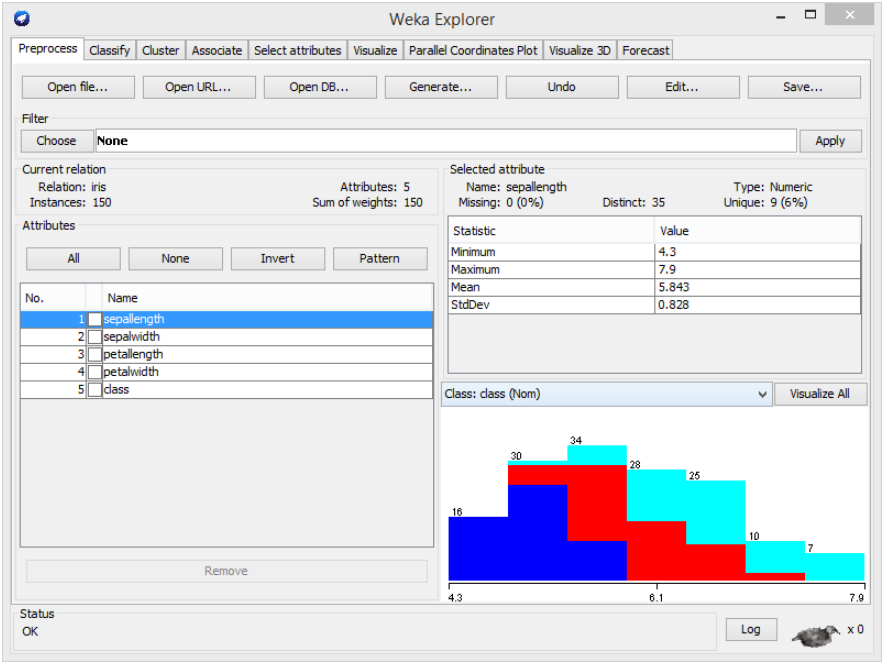



Рисунок А.2 – Вкладка завантаження та попередньої обробки даних

Натискання на кнопку «*Edit*» дозволяє редагувати вихідні дані у табличній формі (з'являється вікно «*Viewer*»).

Панель «*Current relation*» містить 3 значення:

– *Relation* – назва відношення (задачі), яка була зазначена у файлі (застосування фільтрів може змінювати цю назву);

– *Instances* – кількість екземплярів (об'єктів);

– *Attributes* – кількість ознак (атрибутів).

Нижче знаходиться панель «*Attributes*» (атрибути чи іншими словами ознаки об'єктів вибірки даних). На ній знаходяться чотири кнопки і список атрибутів. Атрибути розташовані в тому порядку, в якому вони визначені в файлі даних. Список містить три колонки:

– *No* – номер, що ідентифікує атрибут та використовується для вказівки на атрибут в фільтрах (attributeIndex);

– *Selection tick boxes* - дозволяє вибрати атрибути для подальших операцій (як правило для видалення);

– *Name* – ім'я атрибуту, що визначено у файлі даних.

При виборі одного з атрибутів, його ім'я підсвічується в списку синім кольором. При цьому змінюється вміст правої панелі «*Selected attribute*»:

– *Name* – ім'я атрибуту;

– *Type* – тип атрибуту, наприклад, Nominal або Numeric;

– *Missing* – кількість і відсоток примірників, для яких значення цього атрибуту втрачено (не визначено);

– *Distinct* – кількість різних значень атрибуту;

– *Unique* – кількість і відсоток примірників, для яких значення атрибуту має унікальне значення (такого значення немає у жодного іншого екземпляра).

Нижче цієї статистики відображається інформація про значеннях даного атрибута. Якщо атрибут має тип *nominal* (номінальний або категоріальний, значення з певної множини), список містить усі можливі значення атрибуту з кількістю їх появи. Якщо атрибут відноситься до *numeric* (числовий), то список надає наступну статистичну інформацію: мінімум, максимум, математичне очікування і стандартне відхилення.

Нижче представлена кольорова гістограма значень атрибуту, яка залежно від атрибуту, обраного в якості цільового (що визначає клас), показує розподіл значень атрибуту. Можна подивитися гістограми всіх атрибутів натисканням кнопки «*Visualize all*».

Вище списку всіх атрибутів знаходяться кнопки, що дозволяють вибирати атрибути (за допомогою чекбоксів). Обрані таким чином атрибути можна видалити кнопкою «*Remove*».

Вкладка попередньої обробки даних дозволяє за допомогою фільтрів відібрати і трансформувати дані в необхідний формат. Зліва на панелі «*Filter*» знаходиться кнопка вибору фільтра «*Choose*». Після вибору фільтра з ієрархічного списку його назва з'явиться праворуч від кнопки «*Choose*». Клацнувши по назві фільтра, викликається діалогове вікно налаштування параметрів фільтра. Всі поля даного вікна мають спливаючі підказки. Таке ж діалогове вікно використовується при налаштуванні інших об'єктів програми WEKA (наприклад, класифікаторів). Внизу вікна налаштувань знаходяться 4 кнопки. Перші дві, «*Open...*» та «*Save...*», дозволяють зберегти налаштування об'єкта для майбутнього використання. Кнопка «*Cancel*» дозволяє повернутися без збереження проведених змін. По натисканню кнопки «*OK*» зміни зберігаються, і користувач повертається в «*Explorer window*».

По натисканню на кнопку «*Apply*» у правій частині панелі «*Filter*» відбувається фільтрація (попередня обробка) вихідних даних відповідно до обраного алгоритму. Кнопка «*Undo*» призначена для скасування проведених над даними операцій. Всі зміні, що вносяться в дані в програмі, виконуються в оперативній пам'яті та не впливають на початковий файл. Для збереження змінених даних в новий файлі призначена кнопка «*Save*».

*Зверніть увагу,* що деякі фільтри під час своєї роботи приймають в якості параметру атрибут класу, що задається за допомогою випадаючого списку над гістограмою. Так, наприклад, «*supervised filters*» вимагають, щоб клас був заданий, в той час, коли «*unsupervised attribute filters*» не приймають даний параметр до уваги.

### Класифікація (Classify)

Вкладка «*Classify*» дозволяє вирішувати задачу класифікації для завантаженого набору даних (рис. А.3). Вгорі вкладки знаходиться панель вибору класифікатора «*Classifier*», налаштування параметрів якого подібна вибору фільтра попередньої обробки даних.

На панелі «*Test options*» визначається метод тестування навченого класифікатора:

- на навчальній вибірці (*use training set*);
- на тестовій вибірці з окремого файлу (*supplied test set*);

– по блоках (*cross-validation*);

– за допомогою відсоткового розподілу початкової вибірки на навчання та контроль (percentage split).

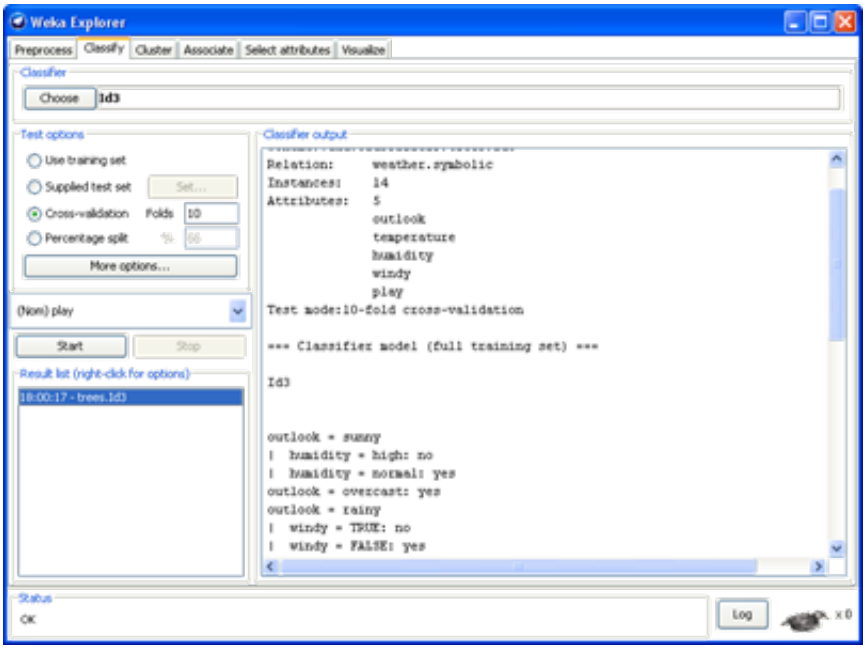



Рисунок А.3 – Вкладка рішення задачі класифікації

При виборі «*cross-validation*» треба вказати, на скільки блоків (фолдів, складок) розбивати вибірку. Наприклад, якщо вказати 10 фолдів, то вибірка буде розділена на 10 рівних частин, потім 9 частин буде використано для навчання і 1 для тестування. Процес буде повторений 10 разів, а результати усереднені. *Leave-one-out* – це спеціальний випадок крос-перевірки, при якому кількість фолдів дорівнює кількості примірників у вибірці. Таким чином, кожен раз тільки один примірник виступає в якості тестової вибірки.

Кнопка «*More options*» дозволяє вибрати вид звіту про навчання класифікатора. Розглянемо параметри, які можна задати:

– *Output model* – відображення побудованої моделі;

– *Output per-class stats* – відображення статистики точність/ефективність та істина/неправда для кожного класу;

– *Output entropy evaluation measures* – відображення оцінки

ентропії на критеріях;

– *Output confusion matrix* – відображення матриці неточностей передбачення класифікатора;

– *Store predictions for visualization* – передбачення класифікатора згруповані таким чином, щоб їх можна було представити в графічному вигляді;

– *Output predictions* – відображення передбачених значень класів для тестової вибірки;

– *Output additional attributes* – використовується у разі якщо разом з передбаченими значеннями класів необхідно вивести додаткові атрибути об'єктів (наприклад, якщо необхідно вивести ID атрибуту для визначення неправильно класифікованих об'єктів);

– *Cost-sensitive evaluation* – помилка класифікації визначається з урахуванням матриці цінності;

– *Random seed for xval/% Split* – визначає випадковий сід (seed), який використовується при рандомізації (перетасуванні) даних перед їх поділом;

– *Preserve order for % Split* – скасовує рандомізацію даних перед їх поділом на навчальну і тестову вибірку;

– *Output source code* – вивід побудованої моделі у вигляді вихідного коду мовою Java, якщо це можливо.

У WEKA класифікатори спроектовані на прогнозування єдиного «класу», який є цільовим атрибутом в задачі класифікації. Деякі класифікатори можуть бути навчені для передбачення категоріальних (nominal) класів (*класифікація*), інші можуть бути навчені тільки для передбачення числових класів (*задача регресії*), а є класифікатори, що вирішують обидва типи задач. За замовчуванням у якості цільового обирається останній атрибут у списку. При необхідності цільовий атрибут можна змінити на панелі «*Test options*».

Процес навчання класифікатора починається натисканням на кнопку «*Start*». Поки класифікатор зайнятий навчанням, маленька пташка в рядку статусу рухається. Процес навчання можна зупинити в будь-який момент натисканням кнопки «*Stop*». Коли процес навчання закінчений, праворуч на панелі «*Classifier output*» з'являється інформація з результатами навчання і тестування, а також заповнюється панель «*Result list*».

Текст на панелі «*Classifier output*» містить інформацію, розбиту на декілька секцій. Розглянемо кожну з них. Секція «*Run information*»

містить параметри схеми навчання, ім'я відношення, екземляри, атрибути і параметри тестування. Секція «*Classifier model (full training set)*» містить текстове представлення моделі класифікації, отриманої в ході навчання. Результати обраної стратегії навчання розбиті на наступні секції:

– *Summary* – статистична інформація, що підводить підсумок точності класифікації для перевірочних примірників;

– *Detailed Accuracy By Class* – більш детальна інформація по кожному класу;

– *Confusion Matrix* – матриця помилок моделі, елементи якої показують кількість тестових екземплярів, чий істинний клас є рядком і чий передбачений клас є колонкою.

Після навчання декількох класифікаторів панель «*Result List*» міститиме кілька записів. Клік лівою кнопкою миші на записі дозволяє переміщатися між результатами аналізу. Праве клацання на записі викликає контекстне меню наступного змісту:

– *View in main window* – показати результати в головному вікні (рівноцінно кліку лівою кнопкою миші);

– *View in separate window* – відкриває незалежне вікно з результатами;

– *Save result buffer* – виводить діалогове вікно, що дозволяє зберегти результати в текстовому файлі;

– *Load model* – завантажує попередньо навчену модель з бінарного файлу;

– *Save model* – зберігає навчену модель в бінарний файл у вигляді серіалізірованого java об'єкта;

– *Re-evaluate model on current test set* – тестує ефективність навченої моделі на наборі даних, завантаженому за допомогою пункту «*Set*» на панелі «*Test option*»;

– *Visualize classifier errors* – виводить вікно з графіком, що візуалізує результати класифікації: вірно класифіковані примірники позначаються хрестиками, а невірно класифіковані – квадратиками;

– *Visualize tree / graph* – надає графічне представлення структури класифікаційної моделі, якщо таке можливо (наприклад, для дерев рішень);

– *Visualize margin curve* – генерує графік, який ілюструє границі передбачення;

– *Visualize threshold curve* – генерує графік, який ілюструє різну

ефективність передбачення, отриману шляхом варіювання порогового значення параметра класифікації;

– *Visualize cost curve* – генерує графік, який дає явне уявлення про очікувані витрати.

Опції неактивні, якщо вони зараз не можуть бути застосовані.

## Кластеризація (Cluster)

Вкладка «*Cluster*» містить схему навчання кластерізаторів (рис. А.4). Вибір функції кластеризації та налаштування її параметрів відбуваються таким же чином, як і для розглянутої вище задачі класифікації.

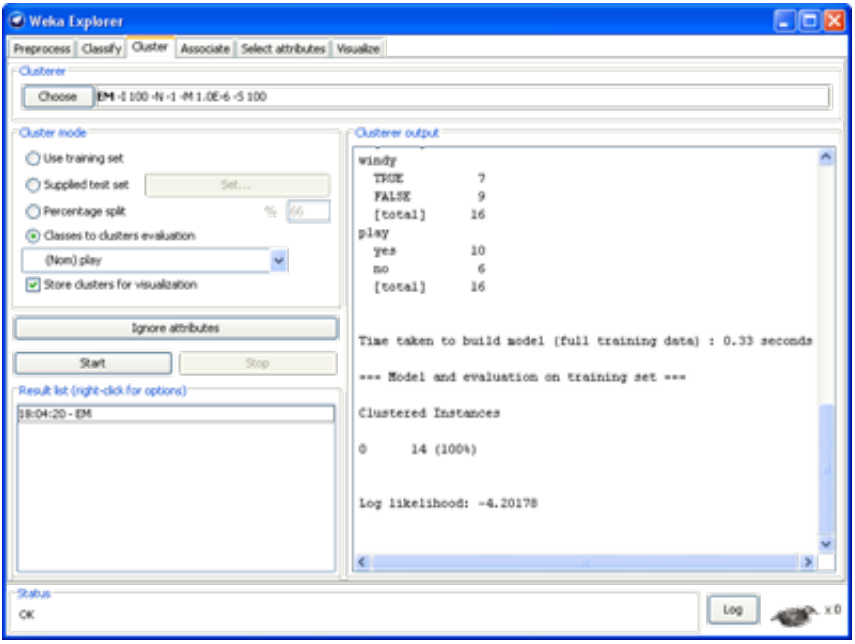



Рисунок А.4 – Вкладка рішення задачі кластеризації

Панель «*Cluster mode*» використовується для того, щоб визначити, що кластеризувати і як оцінювати результати. Перші три опції такі ж, як і для задачі класифікації: *Use training set*, *Supplied test set* и *Percentage split* (при цьому дані використовуються для віднесення до кластеру, а не для передбачення певного класу). Четвертий метод «*Classes to clusters evaluation*» оцінює наскільки добре був обраний

кластер, порівнюючи його з попередньо заданим класом в даних. Додаткова опція у вигляді чекбокса «*Store clusters for visualization*» визначає, чи можливо буде візуалізувати кластери по закінченні навчання. При вирішенні завдань з дуже великими обсягами даних слід відключити дану опцію, щоб уникнути проблем з нестачею пам'яті. Кнопка «*Ignoring Attributes*» дозволяє визначити, які атрибути слід ігнорувати при проведенні кластеризації.

Вкладка Кластеризації також як і вкладка класифікації містить кнопки «*Start/Stop*»**,** панель результатів і список результатів. Клацання правою кнопкою миші на запису у списку результатів дає контекстне меню з двома опціями візуалізації: *Visualize cluster assignments u Visualize tree***.**

### Асоціативні правила (Associate)

Вкладка «*Associate*» містить схему навчання асоціативних правил (рис. А.5). Алгоритми пошуку асоціативних правил обираються, налаштовуються і виконуються так, як описано вище.

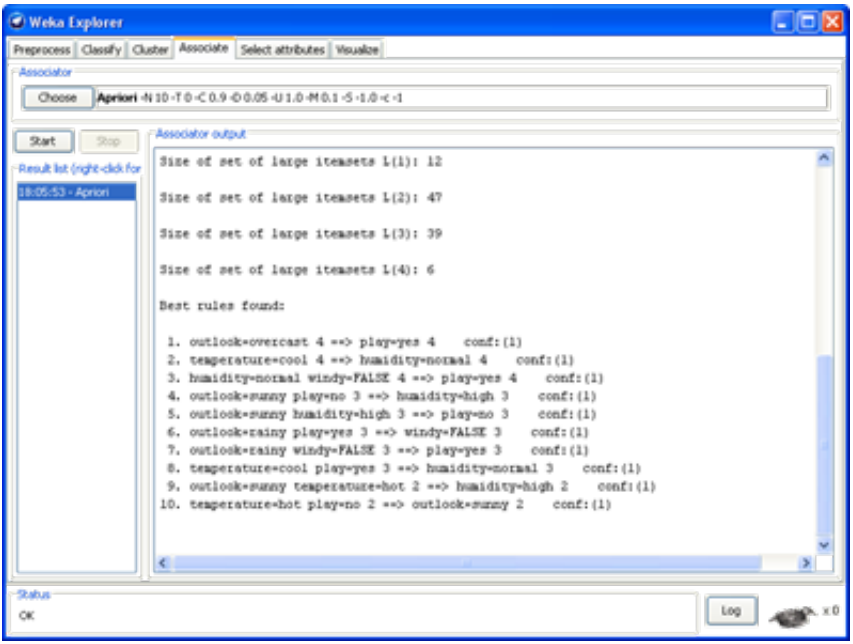



Рисунок А.5 – Вкладка для пошуку асоціативних правил

## Відбір атрибутів (Selecting attributes)

Відбір атрибутів включає перебір всіх можливий комбінацій атрибутів даних для пошуку підмножини атрибутів, що дають найкращий результат передбачення (рис. А.6). Для цього повинні бути налаштовані два об'єкти: оцінювач атрибутів і метод пошуку. Оцінювач (*Atribute Evaluator*) визначає який метод використовується для призначення значущості кожної підмножини атрибутів, а метод пошуку **(***Search Method*) визначає стиль пошуку підмножин.

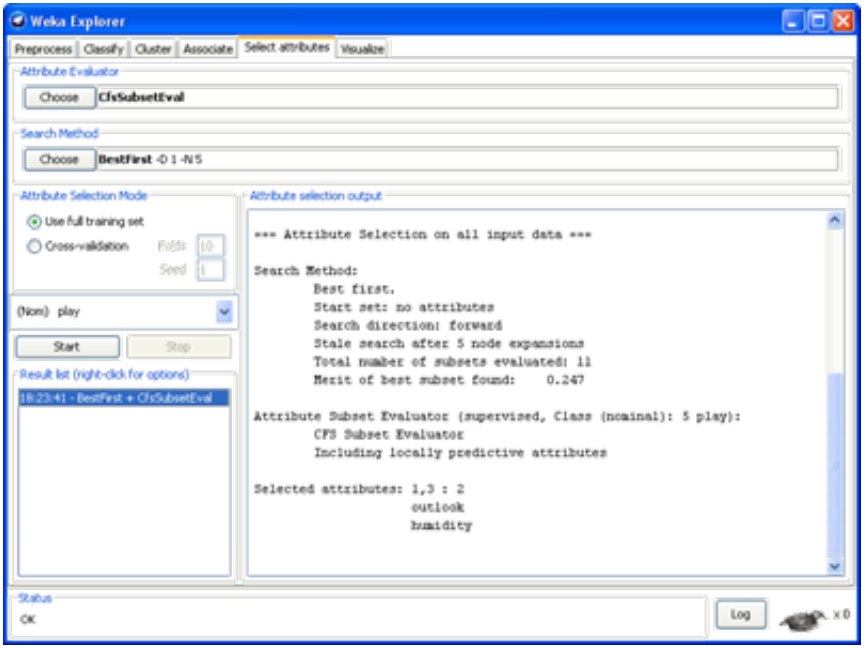



Рисунок А.6 – Вкладка вирішення задачі відбору атрибутів

Панель «*Attribute Selection Mode*» має два параметри:

– *Use full training set* – значимість підмножини атрибутів визначається для повного набору навчальних даних;

– *Cross-validation* (ковзний контроль, крос-перевірка) – значимість підмножини атрибутів визначається за допомогою крос-перевірки. Поля *Fold* і *Seed* визначають кількість блоків (*folds*) і випадковий сид (*seed*), використовуваний при перетасування даних.

Нижче знаходиться список, який задає цільовий атрибут, який буде використовуватися в якості класу.

По натисканню на кнопку «*Start*» запускається процес відбору атрибутів. Коли процес закінчено, результати виводяться на панель результатів «*Attribute selection output*» та додаються до списку результатів. Натискання правої кнопки миші на результати видає контекстне меню. Перші три пункти цього меню **(***View in main window, View in separate window* та *Save result buffer***)** такі ж як і для вкладку класифікації. Додатковими є *Visualize reduced data*, або якщо був обраний метод Principal-Components, то *Visualize transformed data***.**

## Візуалізація (Visualizing)

Вкладка візуалізації дозволяє представити вихідні дані в графічному вигляді (рис. А.7). На ній відображається діаграма розкиду даних для всіх атрибутів з кольоровим кодуванням згідно обраного класу. Розміри кожної з діаграм можуть бути змінені, можуть бути змінені розміри точок. У дані можна додати шум (*jitter*) для виявлення слабких точок. Кожен графік також можна відкрити в окремому вікні натисканням на нього. Для застосування внесених змін та оновлення інформації на графіках необхідно натиснути кнопку «*Update*».

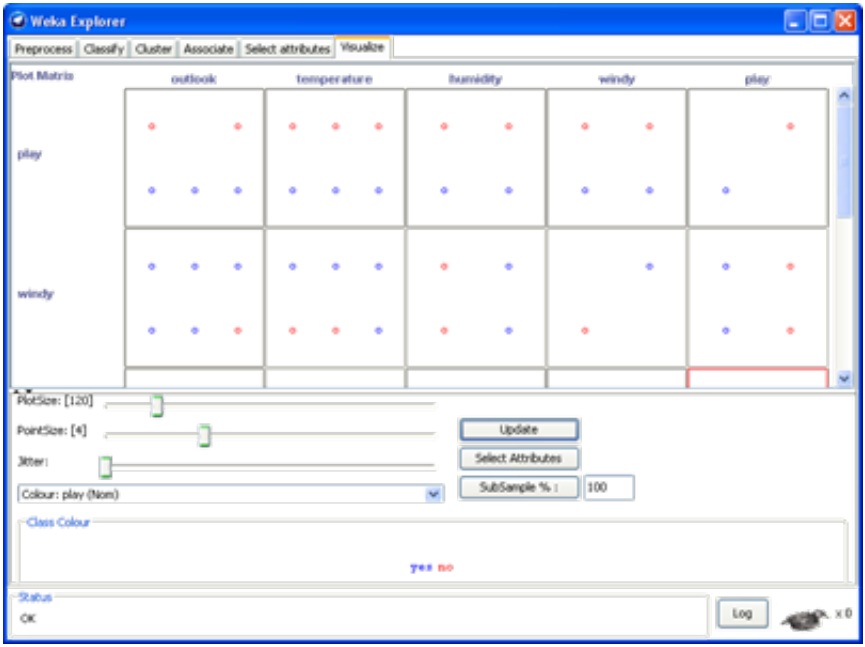



Рисунок А.7 – Вкладка візуалізації вихідних даних

На кожному графіку можна вибрати окремі точки за допомогою випадаючого списку «*Select Instance*» і зберегти їх в окремий файл.

### Модуль Experimenter

Цей модуль дозволяє проводити експерименти: запускати кілька алгоритмів на декількох задачах і отримувати зведений звіт.

На вкладці «*Setup*» (рис. А.8) спочатку активні дві кнопки: «*Open*» (відкрити файл експерименту) і «*New*» (створити новий експеримент), а також дві опції списку параметрів експерименту: «*Simple*» (простий), «*Advanced*» (складний).

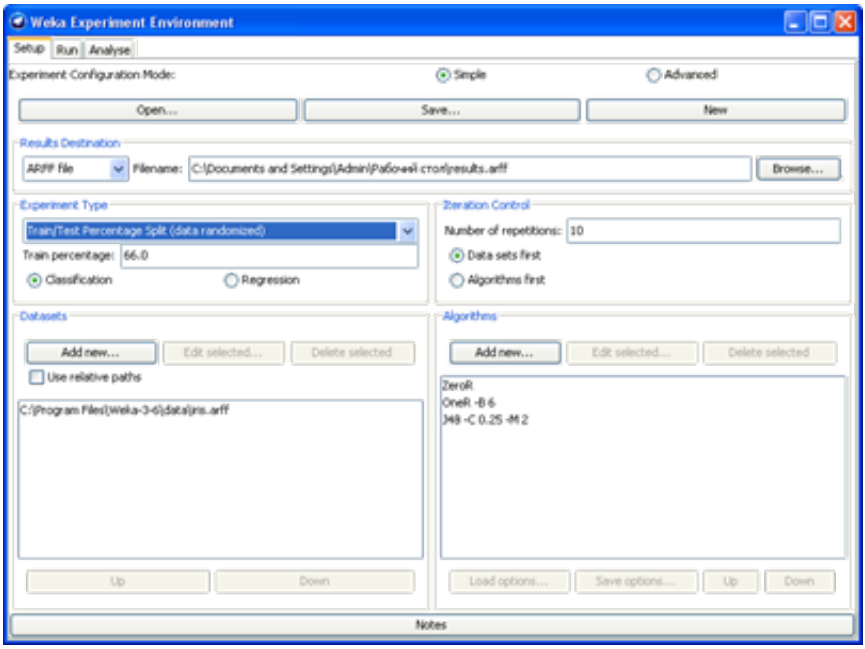



Рисунок А.8 – Установка параметрів експерименту

Створимо новий експеримент. При натисканні на кнопку «*New*» активуються всі опції. На панелі «*Result Destination*» необхідно обрати файл для запису звіту. За замовчуванням файл результатів має розширення arff, тому не затріть файли вихідних даних.

На панелі «*Experiment Type*» обирається тип експерименту:

– *Cross-validation (default)* – проводити верифікацію методом контролю по блоках;

– *Train/Test Percentage Split (data randomized)* – відсотковий поділ вибірки на контроль та навчання з випадковим порядком проходження примірників;

– *Train/Test Percentage Split (order preserved)* – відсотковий поділ вибірки із збереженням порядку (наприклад, коли в одному файлі зберігається і навчальна і тестова вибірка).

На панелі «*Iteration Control*» обирається число ітерацій (повторень експериментів). При повтореннях відбуваються нові розбиття вибірки на навчання і контроль. Також тут вказується порядок проведення експерименту: перебирати спочатку всі задачі або всі алгоритми. На панелі «*Datasets*» можна обрати набори даних, на панелі «*Algorithms*» – алгоритми (параметри їх вибираються аналогічно тому, як це робилося в модулі «*Explorer*»).

На вкладці «*Run*» присутня єдина кнопка «*Start*», що запускає експеримент на виконання.

Вкладка «*Analyze*» (рис. А.9) дозволяє аналізувати результати експериментів. На панелі «*Source*» обирається, який експеримент аналізувати (тільки що проведений чи записаний у файлі).

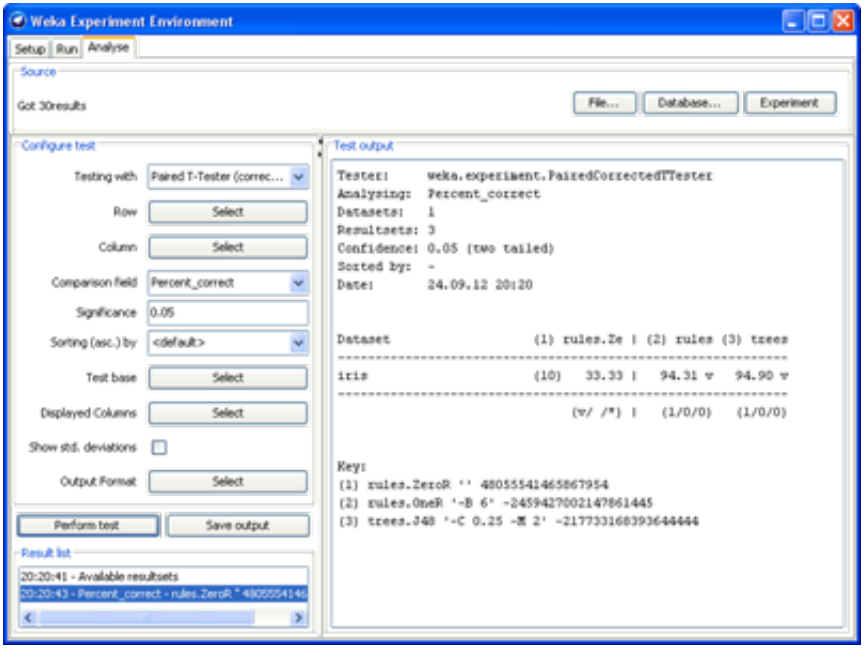



Рисунок А.9 – Оцінка результатів експерименту

На панелі «*Configure test*» визначається вид статистики для аналізу. Кнопка «*Row*» дозволяє вибрати дані, що будуть записані по рядках виведеної матриці (для прикладу виберемо з випадаючого списку *Dataset*), а кнопка «*Column*» – що буде записано за стовпцями (виберемо *Scheme*). У полі «*Comparison field*» вибирається тип даних, якими буде заповнена таблиця (вибір *Percent_correct* забезпечує заповнення відсотками вірної класифікації). Порівняння результатів роботи алгоритмів виконується за допомогою критерію Стьюдента (t-test, Student's tt est). Для генерації звіту натисніть кнопку «*Perform test*».

На рис. А.9. показана статистика роботи трьох алгоритмів на задачі «iris». У колонках статистики вказані методи класифікації, а в рядках – набори даних (задачі). Значення «(10)» на початку рядка, що відповідає набору даних «iris», визначає кількість запусків експерименту (ітерацій, навчання і тестування). Під час експериментів вибірка розбивалася у співвідношенні 66% примірників для навчання і 34% – для тестування.

Відсоток вірно класифікованих примірників для трьох методів показаний у рядку, що відповідає набору даних iris: 33,33% для ZeroR, 94,31% для OneR і 94,90% для J48. Відмітка «v» або «*» означає, що даний результат статистично краще (v) або гірше (*), ніж алгоритм обраний в якості базового (*baseline scheme*, в даному випадку це ZeroR) при зазначеному рівні значущості (*significance*, в даному випадку 0,05). Результати двох алгоритмів OneR і J48 статистично краще, ніж результати базового алгоритму ZeroR.

Внизу кожної колонки знаходиться значення (xx / yy / zz), яке показує кількість разів (за кількістю рядків) у яких результати роботи алгоритму, що знаходиться у назві стовпця, були краще (xx), такі ж (yy) або гірше (zz), ніж результати базового алгоритму на наборах даних, що використовувалися в експерименті.

У даному прикладі був тільки один набір даних і OneR один раз був краще ніж ZeroR і ніколи гірше або еквівалентний йому (1/0/0); J48 також один раз був кращим, ніж ZeroR.

Вибираючи «Number_correct» в якості значення для порівняння (comparison field), в статистиці отримуємо середню кількість вірно розпізнаних примірників для тестової вибірки (з 50 тестових примірників, що є 33% від загальної кількості в 150 екземплярів для набору даних «iris»).

Базовий алгоритм для порівняння може бути змінений за допомогою кнопки «*Test base*».

Вибравши опцію «Summary» в якості бази для порівняння, отримаємо наступну інформацію.

```
      a b c (No. of datasets where [col] >> [row])
      - 1 (1) 1 (1) | a = (1) rules.ZeroR
   0 (0) - 1 (0) | b = (2) rules.OneR
   0 (0) 0 (0) - | c = (3) trees.J48
```

У даному експерименті перший рядок (- 1 1) показує, що метод у колонці b (OneR) краще, ніж метод в рядку a (ZeroR) і метод у колонці c (J48) також краще методу в рядку a. У дужках вказано кількість статистично значущих перемог колонки по відношенню до рядка. Значення 0 показує, що відповідний стовпець не був статистично кращим за рядок.

Вибравши опцію «Ranking» в якості бази для порівняння, отримаємо наступну інформацію:

```
>-< > < Resultset
 1 1 0 trees.J48
 1 1 0 rules.OneR
 -2 0 2 rules.ZeroR
```

Даний тест ранжує методи за загальною кількістю значущих перемог (>) і поразок (<) проти інших методів. Перша колонка (> - <) показує різницю між кількість перемог і поразок.

### Модуль Knowledge flow

Інтерфейс *KnowledgeFlow* відображає концепцію потоків даних (рис. А.10).

Спочатку з примітивів різних видів будується весь шлях, яким мають пройти дані, від джерел до виходів. Зазвичай він такий: джерело даних (*DataSources*) – фільтр (*Filter*) (необов'язковий) – розбиття на навчальну і тестову множину (*Evaluation*) – класифікатор (*Classifiers*) чи кластерізатор (*Clusterers*) чи асоціатор (*Associators*)) – оцінка (*Evaluation*) – виведення даних (*DataSinks*). Практично до будь-якого етапу можна приєднати відображення (*Visualization*).

Примітиви перетягуються в робоче поле з відповідних вкладок (клік на примітив – клік на робоче поле). Входи і виходи примітивів з'єднуються зв'язками, причому не можна з'єднати різнотипні за даними виходи і входи. У примітиву може бути кілька виходів з різними даними. Зв'язки теж бувають різні, залежно від того, які дані по них передаються (з якого виходу вони йдуть). Зв'язуються примітиви шляхом вибору в контекстному меню, що з'являється при натисканні правою кнопкою миші на примітиві в робочому полі, у підпункті меню

*Connection* певного виходу примітиву і наступного кліку на примітив, з яким слід встановити зв'язок. Всі примітиви, доступні для зв'язку, при цьому виділяються.

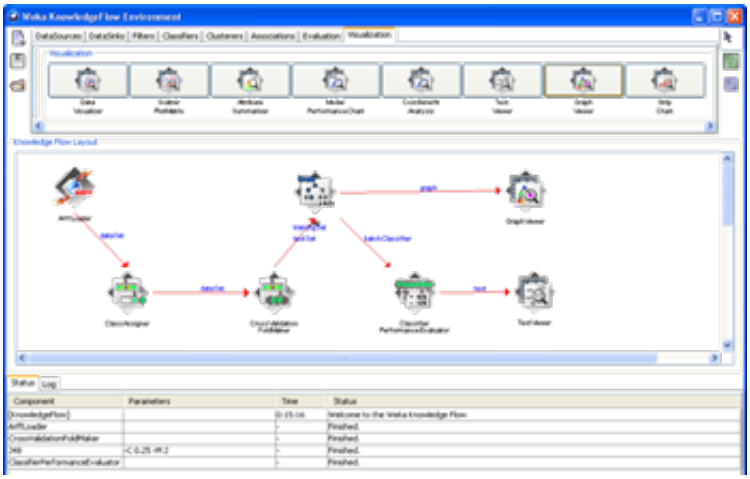


Рисунок А.10 – Модуль Knowledge flow

Після прокладки шляху за допомогою джерела даних завантажують дані, і вони автоматично проходять побудований шлях.

Розглянемо приклад побудови потоку даних на А.10.

1. Відкрийте вкладку джерел даних «*DataSources*» оберіть завантажувач даних «*ArffLoader*» і розмістіть його на робочому полі.

2. Виберіть arff файл для завантаження. Для цього клікніть правою кнопкою миші на завантажувачі, розміщеному на полі, і виберіть зі списку в секції «*Edit*» пункт «*Configure*».

3. Відкрийте вкладку «*Evaluation*», виберіть компонент «*ClassAssigner*» (дозволяє вибрати, який з атрибутів є цільовим) і розмістіть його на полі.

4. Тепер необхідно з'єднати компоненти «*ArffLoader*» та «*ClassAssigner*». Для цього клікніть правою кнопкою миші на завантажувачі «*ArffLoader*» і виберіть з випадаючого списку в секції «*Connections*» пункт «*dataSet*». З'явиться синя лінія, яку необхідно підвести до іконки компоненту «*ClassAssigner*» на полі та клікнути на ньому лівою кнопкою миші. З'явиться красна лінія потоку даних з підписом «*dataSet*», що з'єднує два компоненти.

5. Викличте налаштування компонента «*ClassAssigner*» і виберіть цільовий атрибут (клас).

6. З вкладки «*Evaluation*» виберіть компонент «*CrossValidationFoldMaker*» і розмістіть його на робочому полі. З'єднайте «*ClassAssigner*» і «*CrossValidationFoldMaker*» зв'язком «*dataSet*».

7. З вкладки «*Classifiers*» в секції «*Trees*» виберіть компонент «J48» і розмістіть його на робочому полі. З'єднайте «*CrossValidationFoldMaker*» і «J48» двома зв'язками: «*trainingSet*» і «*testSet*».

8. З вкладки «*Evaluation*» виберіть компонент «*ClassifierPerformanceEvaluator*» і розмістіть його на робочому полі. З'єднайте «*J48*» з цим компонентом зв'язком «*batchClassifier*».

9. З вкладки «*Visualization*» виберіть компонент «*TextViewer*» і розмістіть його на робочому полі. З'єднайте «*ClassifierPerformanceEvaluator*» з цим компонентом зв'язком «*Text*».

10. Запустіть обробку потоку даних за допомогою пункту «*Start loading*» у випадаючому списку завантажувача «*ArffLoader*». Залежно від розміру даних і часу виконання крос-перевірки буде відображена анімація деяких з компонентів в робочій області. Крім того, з'явиться інформація в нижній частині вікна «*Status*»/«*Log*».

11. По закінченні обробки даних перегляньте результати, викликавши пункт «*Show results*» компоненту «*TextViewer*».

12. Крім того, можна підключити компонент «*TextViewer*» та / або «*GraphViewer*» до «*J48*» для того, щоб побачити представлення дерев, побудованих на кожному з фолдів крос-перевірки. Така можливість не передбачена в модулі Explorer.

## Додаток Б
## Варіанти індивідуальних завдань

Обрати з таблиці за номером варіанту (N) з журналу набор даних для дослідження. Дослідити поставлену задачу, характеристики набору даних (атрибути), за необхідності провести попередню обробку даних та зменшити кількість об'єктів у вибірці, виділити аномалії та викиди, обрати стратегію роботи з об'єктами з пропусками, визначити стратегію тестування навчених алгоритмів. Для кожного з алгоритмів провести дослідження їх роботи на поставленій задачі, змінюючи параметри налаштування алгоритму.

Таблиця Б.1 – Набори даних для задачі класифікації

| | | | |
|---|---|---|---|
| 1 | adult.arff | 9 | wine.arff |
| 2 | bank-data.arff | 10 | credit.arff |
| 3 | breast-cancer.arff | 11 | vote.arff |
| 4 | breast-w.arff | 12 | spambase.arff |
| 5 | labor.arff | 13 | zoo.arff |
| 6 | postoperative.arff | 14 | tic-tac-toe.arff |
| 7 | heart-statlog.arff | 15 | mushroom.arff |
| 8 | diabetes.arff | 16 | vehicle.arff |

Для вирішення задачі регресії обрати одну задачу за номером варіанту (N mod 7)+1. Іншу задачу обрати на власний смак.

Таблиця Б.2 – Набори даних для задачі регресії

| | | | |
|---|---|---|---|
| 1 | cpu.arff’ | 5 | housing.arff |
| 2 | auto_mpg.arff | 6 | bodyfat.arff |
| 3 | winequality-red.csv | 7 | fishcatch.arff |
| 4 | autoprice.arff | 8 | auto93.arff |