УДК 61:004.651(075.8)
І.Є.Андрущак
Луцький національний технічний університет

РОЗРОБКА КЛІНІЧНОЇ ЕКСПЕРТНОЇ СИСТЕМИ МЕТОДОМ ІНДУКЦІЇ ДЕРЕВА РІШЕНЬ

І.Є.Андрущак. Розробка клінічної експертної системи методом індукції дерева рішень. У роботі розглядаються питання програмної реалізації методу індукції дерева рішень на основі інформаційних показників. Такий підхід дозволяє розробити систему підтримки клінічних рішень. Проект реалізовано в середовищі Netbeans на основі Java-класів. Показано використання SQL-запитів для обчислення інформаційних показників.
Ключові слова: прийняття рішень, дерево рішень, Java, SQL.

И.Е.Андрущак.Разработка клинической экспертной системы методом индукции дерева решений. В работе рассматриваются вопросы программной реализации метода индукции дерева решений на основе информационных показателей. Такой подход позволяет разработать систему поддержки клинических решений. Проект реализован в среде Netbeans на основе Java-классов. Показано использование SQL-запросов для вычисления информационных показателей.
Ключевые слова: принятие решений, дерево решений, Java, SQL.

I.Ye.Andruschak.Clinical development of expert system the induction of decision trees. The paper deals with the software implementation of the method of induction of decision trees based on the information provided. This approach allows you to develop a system to support clinical decisions. The project was implemented in an environment based on Netbeans Java-classes. Shown using SQL-query to compute the information provided.
Keywords: decision making, decision tree, Java, SQL.

Вступ. В медицині поняття «диференціальної діагностики» означає системний підхід, що грунтується на доказовості, для визначення причини симптомів, що спостерігаються, у випадку, коли є кілька альтернативних пояснень, а також для зменшення переліку можливих діагнозів.

Сьогодні медичне діагностування може виконуватися автоматично з використанням комп'ютеризованих систем та алгоритмів. Такі системи переважно називаються діагностичними системами підтримки прийняття рішень або медичними діагностичними системами. Вони належать до загальнішого класу клінічних систем підтримки прийняття рішень [Марценюк, 2004-2012]. Метою таких систем є системний супровід лікаря в процесі диференціальної діагностики. Багато з таких систем можуть надавати результати навіть коли не вистачає даних, тобто в умовах невизначеності, і що найважливіше – вони не обмежені щодо кількості інформації, яку можуть зберігати та обробляти.

Одним з підходів, що відображає природній процес мислення при диференціальній діагностиці, є метод індукції дерева рішень. Протягом кінця 1970-х на початку 1980-х років J.R.Quinlan [Quinlan, 1986] розробив алгоритм побудови дерева рішень ID3 (ітеративний дихотомайзер). Пізніше J.R.Quinlan представив алгоритм C4.5 (наступник ID3), який став еталоном, з яким часто порівнюються новітні алгоритми в галузі машинних знань. У 1984 році група статистиків (L.Breiman, J.Friedman, R.Olshen, C.Stone) опублікували роботу щодо Classification and Regression Trees (CART) [Breiman, 1984], в якій описали побудову бінарних дерев рішень. Алгоритми ID3 та CART, незважаючи на те, що були розроблені незалежно і приблизно у той же час, реалізують подібний підхід до навчання дерев рішень на основі навчальних даних. При цьому дерева рішень будуються в результаті рекурсивної процедури типу «зверху-вниз». Більшість алгоритмів індукції дерев рішень також відповідають цьому загальному підходу. При цьому навчальна множина рекурсивно поділяється на менші підмножини по мірі того, як будується дерево.

Математично задача індукції дерева рішень формулюється таким чином. Маємо множину D, що містить N наборів навчальних даних. При цьому кожен i-й набір $(A_1^i, A_2^i, ..., A_p^i, C^i)$ складається з вхідних даних – атрибутів $A_1, ..., A_p$ та вихідних даних – атрибуту класу C. Атрибути $A_1, ..., A_p$ можуть приймати як чисельні, так і категоріальні значення. Атрибут класу C приймає одне з K дискретних значень: $C \in \{1, ..., K\}$. Метою є прогнозування деревом рішень значення атрибуту класу C на основі значень атрибутів $A_1, ..., A_p$. При цьому слід максимізувати

точність прогнозування атрибуту класу, а саме $P\{C = c\}$ на термінальних вузлах для довільного $c \in \{1,...,K\}$. Алгоритми індукції дерев рішень автоматично розбивають на вузлах значення чисельних атрибутів A_i на два інтервали: $A_i \le x_i$ та $A_i > x_i$, а категоріальних атрибутів A_j – на дві підмножини: $A_j \in S_j$, $A_j \notin S_j$. Розбиття чисельних атрибутів грунтується, як правило, на мірах на основі ентропії, або індексі Джині [Han, 2001]. Процес розбиття рекурсивно повторюється до тих пір, поки не спостерігатиметься покращення точності прогнозування. Останній крок включає вилучення вузлів для уникнення оверфітінгу моделі. В результаті ми повинні отримати множину правил, що йдуть від кореня до кожного термінального вузла, містять нерівності для чисельних атрибутів та умови включення для категоріальних атрибутів.

Метою роботи є розробити метод індукції дерева рішень з можливістю програмної реалізації у вигляді клінічної експертної системи.

Метод індукції дерева рішень. За основу взято таку рекурсивну процедуру роботи [Han, 2001].

Генерація дерева рішень

Вхідні дані: D – множина навчальних наборів даних $(A_1^i, A_2^i,..., A_p^i, C^i)$.

Вихідні дані: дерево рішень

Метод:

1. Створити вузол N.

2. Якщо усі набори в D належать до спільного класу C, тоді повернути вузол N як листок із назвою класу C.

3. Якщо список атрибутів (а отже і D) є порожнім, тоді повернути вузол N як листок із назвою найпоширенішого класу в D.

4. Застосувати *Алгоритм відбору атрибуту* із списку атрибутів і для множини D з метою відшукання «найкращого» атрибуту поділу.

5. Вилучити атрибут поділу із списку атрибутів.

6. Для кожної умови поділу j для атрибуту поділу розглянемо D_j – множину наборів з D, що задовольняють умову поділу j.

7. Якщо D_j – порожня, тоді приєднати до вузла N листок під заголовком найпоширенішого класу в D, інакше – приєднати до N вузол, що повертається рекурсивним викликом методу *Генерація дерева рішень* з вхідними даними D_j та список атрибутів.

8. Кінець циклу кроку 6.

9. Повернути вузол N.

В основу *Алгоритму відбору атрибуту* на j - му кроці рекурсії покладено такий інформаційний показник:

$$Gain(A_i) = Info(D_j) - Info_{A_i}(D_j). \quad (1)$$

Тут

$$Info(D_j) = -\sum_{k=1}^{K} p_k^j \log_2(p_k^j) \quad (2)$$

– інформація, потрібна для класифікації набору $(A_1, A_2,..., A_p)$ в D_j,

$$Info_{A_i}(D_j) = \sum_{l=1}^{K_i} \frac{\#(D_j^l)}{\#(D_j)} Info(D_l) \quad (3)$$

– інформація, потрібна для класифікації $(A_1, A_2,..., A_p)$ в D_j після поділу D_j на підмножини D_j^l відповідно до значень атрибуту A_i.

У формулі (2) ймовірність того, що довільний набір з D_j належить множині C_{k,D_j} оцінюється як $p_k^j = \frac{\#(C_{k,D_j})}{\#(D_j)}$, де C_{k,D_j} – множина наборів з D_j, для яких атрибут класу $C = k$. Тут $\#(\bullet)$ – кількість елементів в множині.

У формулі (3) $\frac{\#(D_j^l)}{\#(D_j)}$ – оцінка ймовірності того, що довільний набір з D_j належить множині D_j^l, де D_j^l – множина наборів з D_j, для яких атрибут $A_i = a_i^l$. Тут атрибут $A_i \in \{a_i^1, a_i^2, ..., a_i^{K_i}\}$.

Отже, $Gain(A_i)$ оцінює зменшення інформації, необхідної для класифікації довільного набору даних в D_j за рахунок відомого значення атрибуту A_i. Таким чином з наявних атрибутів на кожному вузлі дерева рішень для умови поділу слід відбирати атрибут A_{i^*} з найбільшим значенням $Gain(A_{i^*})$. В результаті такого вибору для завершення процесу класифікації набору даних в D_j вимагатиметься найменше інформації.

Програмна реалізація. Метод реалізовано в середовищі розробки Netbeans на мові програмування Java. Базу навчальних даних розгорнуто на сервері MySQL. На рис.1 представлено концептуальну модель інформаційної системи. У класі DecisionTree безпосередньо реалізовано метод індукції дерева рішень. У клас DataManager надходять виклики від DecisionTree на виконання запитів до бази даних mysql щодо отримання навчальних даних.

База даних mysql складається з двох таблиць – таблиці attribute, призначеної для зберігання інформації про атрибути та таблиці categorized_data – для наборів навчальних даних. Структура таблиць на мові SQL для прикладу наведена нижче:

```
CREATE TABLE mysql.attribute (
                id integer not null unique,
                attribute_name varchar(25),
                attribute_field_name varchar(25),
        primary key (id)
   ) ENGINE=InnoDB;
CREATE TABLE mysql.categorised_data (
                id integer not null unique,
                A1 varchar(12),
                A2 varchar(8),
                A3 varchar(7),
                A4 varchar(7),
                A5 varchar(7),
                class varchar(8),
        primary key (id)
   ) ENGINE=InnoDB;
```

Програмні класи проекту включено до пакету decision_tree.model. Сюди входять: beans-класи Attribute, Attribute_for_list та CategorisedData для роботи з даними відповідних таблиць. SQL-запити щодо отримання відповідних даних, включаючи розрахунки інформаційних показників реалізовано в класі AttributeListPeer.

Клас DecisionTree є нащадком класу DefaultTreeModel пакету javax.swing.tree. Він має два елементи класу: m_dataManager – менеджер даних та m_htAttribute_list – хеш-таблиця із списком атрибутів. Хеш-таблиця із списком атрибутів (у методах класу DecisionTree виступає під назвою htAttribute_list) створюється для кожного вузла дерева рішень.

Вона має два призначення – поряд із списком включених для даного вузла атрибутів зберігати умови поділу (splitting conditions), які перейшли до даного вузла від вузлів-батьків.

Кожен вузол дерева рішень є об'єктом класу DefaultMutableTreeNode. В якості об'єкта кожен вузол зберігає об'єкт класу NodeObject, декларація якого наведена нижче:

```
class NodeObject {
   Attribute attribute;
   Hashtable htAttribute_list;
   String splitting_criterion;
   String sLabel;
   public String toString() {
      if (splitting_criterion.matches("")) { return sLabel; }
      else return "if '" + splitting_criterion + "' then '" + sLabel + "'"; }
}
```

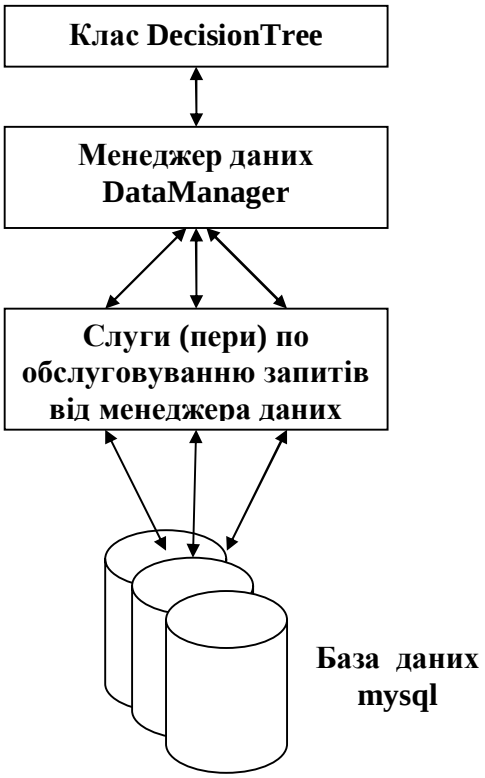

Рис.1. Концептуальна модель інформаційної системи індукції дерева рішень

Тут attribute – атрибут, який повертається методом Attribute_selection_method, splitting_criterion – умова поділу, яка переходить від батьківського вузла, sLabel – надпис на вузлі. Хеш-таблиця htAttribute_list використовується для побудови наборів навчальних даних D_j для кожного із вузлів і має таку структуру:

Тип ключа	int
Тип об'єкта	Attribute_for_list
Структура об'єкта	Attribute attribute; Hashtable htSplitting_outcomes; String splitting_criterion; boolean included;

Тут included – булева змінна-прапорець належності атрибуту attribute до списку атрибутів даного вузла. Можна показати, що коли included=true, то вузол з назвою attribute є для даного

вузла дочірнім (на певному нижчому рівні ієрархії). У випадку, коли атрибут attribute не входить до списку атрибутів для даного вузла (included=false), то вузол з назвою attribute є батьківським (на певному рівні ієрархії), а в змінній splitting_criterion зберігається умова поділу, якій підлягає даний вузол відносно батьківського вузла attribute.

Хеш-таблиця htSplitting_outcomes містить усі можливі наслідки (умови поділу) щодо атрибуту attribute.

Метод Generate_decision_tree є безпосередньою реалізацією методу індукції дерева рішень. Заголовок методу має вигляд:

```
private DefaultMutableTreeNode Generate_decision_tree (Hashtable htAttribute_list,
DefaultMutableTreeNode dmtnSubroot, String splitting_criterion)
```

В якості аргументів метод використовує кореневий вузол дерева, список пов'язаних з ним атрибутів htAttribute_list та умову поділу splitting_criterion. В якості значення метод повертає дочірній вузол типу DefaultMutableTreeNode. Шляхом рекурсивного виклику методу Generate_decision_tree будується дерево рішень.

З метою візуалізації представлення дерева використано клас javax.swing.JTree. При цьому дерево рішень створюється виводиться за допомогою операторів:

```
dtDecision_tree = new DecisionTree(dmtnRoot, dataManager, htAttribute_list);
jTree1.setModel(dtDecision_tree);
```

SQL-реалізація розрахунку інформаційних показників. Ключовим в реалізації методу Attribute_selection_method є розрахунок інформаційних показників $Info(D_j)$ та $Info_{A_i}(D_j)$ на j-му кроці рекурсії для атрибута A_i. Так показник $Info(D_j)$ розраховується методом:

```
public double getInfo$D$ (DataManager dataManager, Hashtable htAttribute_list).
```

Зауважимо, що множина наборів D_j тут описується хеш-таблицею htAttribute_list, з якої отримуємо перелік включених атрибутів sAttribute_list та умов поділу sConditions.

Мова структурованих запитів SQL має досить гнучкі засоби для реалізації алгоритмів в галузі машинних знань [??]. Так використавши вкладені запити, псевдоніми та агрегативні функції можна розрахувати $Info(D_j)$ в результаті виконання такого SQL-запиту:

```
String sql = "select -SUM((Alias1.Ci/Alias2.D)*(LOG(Alias1.Ci/Alias2.D)/LOG(2))) from " +
"(select SUM(1) as Ci from (select " + sAttribute_list + ",class from categorised_data " +
(sConditions.matches("")?"": " where " + sConditions) +  ")Alias3 group by Alias3.class)Alias1, " +
"(select SUM(1) as D from (select " + sAttribute_list + " from categorised_data " +
(sConditions.matches("")?"": " where " + sConditions) + ")Alias4)Alias2";
```

Показник $Info_{A_i}(D_j)$ розраховується методом:

```
public static double getInfo_A$D$(DataManager dataManager, int i, Hashtable htAttribute_list)
```

Остаточно $Info_{A_i}(D_j)$ обчислюється в результаті виконання такого SQL-запиту:

```
String sql = "select SUM((Alias1.Dj/Alias2.D)*Alias3.Info$Dj$) from " + "(select SUM(1) as Dj from
(select * from categorised_data " + (sConditions.matches("")?"": " where " + sConditions) + ")Alias6
group by Alias6." + ((Attribute_for_list)htAttribute_list.get(A)).attribute.getAttributeFieldName() +
")Alias1, " + "(select SUM(1) as D from (select " + sAttribute_list + " from categorised_data " +
(sConditions.matches("")?"": " where " + sConditions) + ")Alias7)Alias2, " + "(select -
SUM((Alias4.Ci/Alias5.D)*(LOG(Alias4.Ci/Alias5.D)/LOG(2))) as Info$Dj$ from  " + "(select SUM(1)
as Ci from (select " + sAttribute_list + ",class from categorised_data " + (sConditions.matches("")?"": "
where " + sConditions) + ")Alias8 group by Alias8.class)Alias4, " + "(select SUM(1) as D from (select "
+ sAttribute_list + " from categorised_data " + (sConditions.matches("")?"": " where " + sConditions) +
")Alias9)Alias5)Alias3";
```

Приклад. Метою є побудувати дерево рішень щодо діагностування серцевої недостатності на основі даних 63-х пацієнтів навчально-практичного центру первинної медико-санітарної

допомоги Тернопільського державного медичного університету імені І.Я.Горбачевського. Використано таку таблицю атрибутів:

INSERT INTO mysql.attribute (id, attribute_name, attribute_field_name) VALUES (1, 'What is age?', 'A1'), (2, 'What is sex?', 'A2'), (3, 'What is pulse?', 'A3'), (4, 'What is SAP?', 'A4'), (5, 'What is DAP?', 'A5');

Набори включають лише категоріальні дані (попередньо оброблені), наприклад:

INSERT INTO mysql.categorised_data (id, A1, A2, A3, A4, A5, class) VALUES (1,'senior','female','normal','high','high','healthy');

На рис.2 представлене побудоване дерево рішень. Час, затрачений на індукування дерева – 860 мілісекунд.

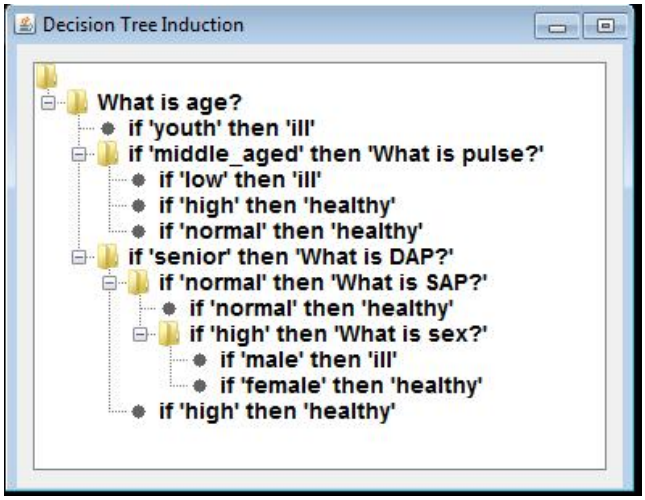

Рис.2. Дерево рішень для діагностування серцевої недостатності

Висновки. У роботі розглянуто питання програмної реалізації методу індукції дерева рішень на основі інформаційних показників.

На прикладі продемонстровано, що такий підхід дозволяє розробити систему підтримки клінічних рішень.

Показано, що мова SQL має достатні синтаксичні можливості, що дозволяють розрахувати інформаційні показники на основі таблиць баз даних.

За рахунок використання Java-класів дана реалізації методу індукції дерева рішень є веб-інтегрованою.

Перспективами досліджень є аналіз продуктивності програмного продукту залежно від кількості атрибутів та обсягу наборів навчальних даних.

Список використаних джерел.

1. J.Han and M.Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, San Francisco, 1st edition, 2001.
2. T.Hastie, R.Tibshirani and J.H.Friedman, The Elements of Statistical Learning, Springer, New York, 1st edition, 2001.
3. C.Ordonez, Comparing association rules and decision trees for disease prediction, In Proc. ACM HIKM Workshop, 2006, pp. 17-24.
4. C.Ordonez, Integrating K-means clustering with a relational DBMS using SQL, IEEE Transactions on Knowledge and Data Engineering (TKDE) 18(2) (2006), 188-201.
5. J.R.Quinlan. Induction of decision trees. Machine Learning, 1: 81-106, 1986.
6. J.R.Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
7. L.Breiman, J.Friedman, R.Olshen, and C.Stone. Classification and Regression Trees. Wadsworth International Group, 1984.
8. Марценюк В.П., Кравец Н.О. О программной среде проектирования интеллектуальных баз данных // Клиническая информатика и телемедицина – 2004. – №1. – с.47-53.
9. Марценюк В.П. Математичні моделі в системі підтримки прийняття рішень страхового забезпечення лікування онкологічних захворювань: підхід на основі динаміки Гомперца / В.П. Марценюк, І.Є. Андрущак, І.С. Гвоздецька, Н.Я. Климук // Доповіді Національної академії наук України. –2012. – №10. – С. 34-39.
10. Марценюк В.П. Підхід на основі актуарних математичних моделей до задач страхової медицини / В. П. Марценюк, І.Є. Андрущак, Н.Я Климук // Медична інформатика та інженерія. Науково-практичний журнал. – 2010. – №4. – С.85-87.