МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Методичні вказівки
до лабораторних робіт з дисципліни

«ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ»

для студентів усіх форм навчання
спеціальності 121 - Інженерія програмного забезпечення

ЗАТВЕРДЖЕНО
кафедрою «Програмної інженерії».
Протокол № 2 від 25.09.2023р.

ХАРКІВ – 2023 р.

Методичні вказівки до лабораторних робіт з дисципліни «Інтелектуального аналізу даних» для студентів усіх форм навчання спеціальності 121 – Інженерія програмного забезпечення./ Упоряд. І.В. Афанасьєва, К.Г. Онищенко – Харків: ХНУРЕ, 2023. – 131 с.

Упорядник Ірина АФАНАСЬЄВА, Костянтин ОНИЩЕНКО

Рецензент: Олег КОБИЛІН, к. т. н, доц. каф. «ІНФ»

ЗМІСТ

ВСТУП

Засоби сучасної інформаційної технології в останній час уможливили накопичення і зберігання великих обсягів даних про бізнесові процеси. Ці дані можуть знаходитися в корпоративних базах або сховищах даних. Вони містять важливі закономірності і зв’язки між системними характеристиками, які можуть бути використані для прийняття обгрунтованих управлінських рішень.

Наразі виникла проблема розробки методів відкриття таких закономірностей, про існування яких користувачі можуть і не знати. Проте традиційний аналіз даних передбачає введення даних в стандартні або налаштовані користувачем моделі, тобто в будь-якому випадку допускається, що зв'язки між різними показниками добре відомі і можуть бути виражені математично.

Однак, в багатьох випадках зв'язки не можуть бути апріорі відомі. У таких ситуаціях моделювання стає неможливим і тут можна застосовувати датамайнінг (*Data Mining) – інтелектуальний аналіз даних (ІАД).* Тому, особливо важливим аспектом підготовки магістрів за спеціальністю «Інжеренія програмного забезпечення» є успішне засвоєння ними дисципліни "Інтелектуальний аналіз даних”.

У результаті вивчення дисципліни “Інтелектуальний аналіз даних” студент повинен :

Знати:

– сутність та призначення Data Mining;

– характеристики процесів та активностей дейтамайнінгу;дерево методів дейтамайнінгу; доступне програмне забезпечення ІАД;

– призначення та основні характеристики генетичних алгоритмів і програмних агентів;

– вивчити основні методи генетичного пошуку.

Вміти:

– використовувати генетичні методи для розв’язку оптимізаційних задач, будувати дерево методів дейтамайнінгу; проводити кластерний аналіз засобами дейтамайнінгу;

– здійснювати вибір відповідних логічних методів із побудовою таблиці трансакцій; будувати крос-таблицю;

– вміло застосовувати доступне програмне забезпечення дейтамайнінгу. Також використовувати різноманітні пакети для опрацюваннння даних.

Методичні вказівки до виконання лабораторних робіт з дисципліни “Інтелектуальний аналіз даних” включають 4 теоретичних розділи та 4 лабораторних роботи, кожний із яких містить необхідний методичний матеріал для вивчення даного предмету.

1 ЛАБОРАТОРНА РОБОТА № 1. МЕТОДИ МОДЕЛЮВАННЯ ТА ПРОГНОЗУВАННЯ. ПРОСТИЙ ТА МНОЖИННИЙ РЕГРЕСІЙНИЙ АНАЛІЗ. ПЕРЕВІРКА ПЕРЕДУМОВ ТА ПРИПУЩЕНЬ ВИКОРИСТАННЯ РЕГРЕСІЙНОГО АНАЛІЗУ.

1.1 Мета роботи

Ознайомитись та отримати навички опрацювання даних використовуючи лінійну, багатофакторну та нелінійну регресії у пакеті Statistica.

1.2 Методичні рекомендації до самостійної роботи студентів

У статистиці лінійна регресія — це метод моделювання залежності між скаляром y та векторною (у загальному випадку) змінною X. У випадку, якщо змінна X також є скаляром, регресію називають простою.

При використанні лінійної регресії взаємозв'язок між даними моделюється за допомогою лінійних функцій, а невідомі параметри моделі оцінюються за вхідними даними. Подібно до інших методів регресійного аналізу лінійна регресія повертає розподіл умовної імовірності y в залежності від X, а не розподіл спільної імовірності y та X, що стосується області мультиваріативного аналізу.

При розрахунках параметрів моделі лінійної регресії як правило застосовується метод найменших квадратів, але також можуть бути використані інші методи. Так само метод найменших квадратів може бути використаний і для нелінійних моделей. Тому МНК та лінійна регресія хоч і є тісно пов'язаними, але не є синонімами.

Лінійний парний регресійний аналіз

Моделі та методи регресійного аналізу займають центральне місто у математичному апараті економетрії. Задачею регресійного аналізу являється

установлення форми залежності між змінними, оцінка функції регресії, прогноз значень залежної змінної.

Функціональна, статистична та кореляційна залежності

У природознавчих науках часто йде мова о функціональній залежності, коли кожному значенню однієї змінної відповідає цілком певне значення іншої.

В економиці у багатьох випадках між змінними існують залежності, коли кожному значенню однієї змінної відповідає не деяке певне, а множина можливих значень іншої змінної. Іначе кажучи, кожному значенню однієї змінної відповідає певне (умовне) розподілення іншой змінної. Така залежність отримала назву статистичної (або імовірної,стохастичної).

Виникнення поняття статистичної залежності обумовлюється тим, що залежна змінна підпадає під вплив неконтролюємих або неврахованих факторів, а також тим, що вимірювання значень змінних неминуче супроводжується декотрими випадковими похибками.

В силу невизначенності статистичной залежності між X та Y для дослідження представляє інтерес усереднена по X схема залежності. Тобто закономірність у вимірюванні умовного математичного сподівання Mx(y).

Якщо залежність між двома змінними така, що кожному значенню однієї змінної відповідає певне умовне математичне сподівання іншої, то така статистична залежність називається кореляційною:

$$Mx(y)=F(x).$$

У регресійному аналізі розглядаються залежність випадкової змінної *Y* від однієї (або декількох) невипадкової незалежної змінної *X*. Така залежність може виникнути у випадку, коли при кожному значенні змінної *X* відповідні значення *Y* підпадають під вплив неконтролюємих факторів. Така залежність *Y* від *X* (іноді її називають регресійною) також може бути представлена у вигляді модельного рівняння рівняння регресії.

Y - функція відгуку, пояснювальна, ендогенна, результативна ознака, вихідна, результативна; *X* - пояснююча, екзогенна, предикторна, фактор, регресор, факторний признак.

Лінійна парна регресія

Якщо за розташуванням точок даних можна припустити наявність лінійної регресіїної моделі

$$Y=\beta_0+\beta_1 X+\varepsilon, \qquad (1)$$

або

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, i = \overline{1,n},$$

то рівняння регресії шукається у вигляді лінійного рівняння

$$\hat{y}=b_0+b_1 x, \qquad (2)$$

де $\hat{y}$ - це оцінка $M_x(y)$, b_0 - оцінка β_0, b_1 - оцінка β_1.

Згідно методу найменших квадратів (1МНК) невідомі параметри b_0 та b_1 обираються таким чином, щоб сума квадратів відхилень емпіричних значень y_i від теоретичних значень $\hat{y}_i$ була найменшою:

$$S = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} (b_0 + b_1 x_i - y_i) \to \min$$

За необхідними умовами екстремуму:

$$\begin{cases} \frac{\partial s}{\partial b_0} = 2\sum_{i=1}^{n}(b_0 + b_1 x_i - y_i) = 0, \\ \frac{\partial s}{\partial b_1} = 2\sum_{i=1}^{n}(b_0 + b_1 x_i - y_i)x_i = 0. \end{cases}$$

Відкіля після перетворень отримаємо систему нормальних рівнянь:

$$\begin{cases} b_0 n + b_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i, \\ b_0 \sum_{i=1}^{n} x_i + b_0 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i; \end{cases}$$

$$\begin{cases} b_0 + b_1 \overline{x} = \overline{y}, \\ b_0 \overline{x} + b_1 \overline{x^2} = \overline{xy}; \end{cases}$$

де

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}, \overline{x^2} = \frac{\sum_{i=1}^{n} x_i^2}{n}, \overline{xy} = \frac{\sum_{i=1}^{n} x_i y_i}{n}$$

Після розв'язання останньої системи отримуємо коефіцієнт регресії *Y* по *X*:

$$b_1 = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2} \qquad (3)$$

Коефіцієнт регресії *Y* по *X* показує, наскільки одиниць у середньому змінюються змінна *Y* при збільшені змінної *X* на одну одиницю.

Формула обчислення параметру b_0:

$$b_0 = \bar{y} - \bar{b}_1\bar{x}. \qquad (4)$$

Для оцінки щільності кореляційного зв'язку використовується *коефіцієнт кореляції*:

$$r = \frac{\overline{xy} - \bar{x}\bar{y}}{S_x S_y},$$

$$S_x = \sqrt{\overline{x}^2 - \bar{x}^2}, S_y = \sqrt{\overline{y}^2 - \bar{y}}^2, \qquad (5)$$

де S_x, S_y - середньоквадратичні відхилення.

Властивості коефіцієнта кореляції:

1. Коефіцієнт кореляції приймає значення на відрізку *[-1;1]*, тобто $-1 \le r \le 1$. Чим ближче $|r|$ до одиниці, тим тісніше зв'язок.

Дві кореляційні залежності наведені на рис. 1.1. Очевидно, що у випадку *а* залежність між змінними менш щільна, і коефіцієнт кореляції повинен бути менш, ніж у випадку *б*, так як точки кореляційного поля *а* подальш відстоять від лінії регресії, ніж точки поля *б*.

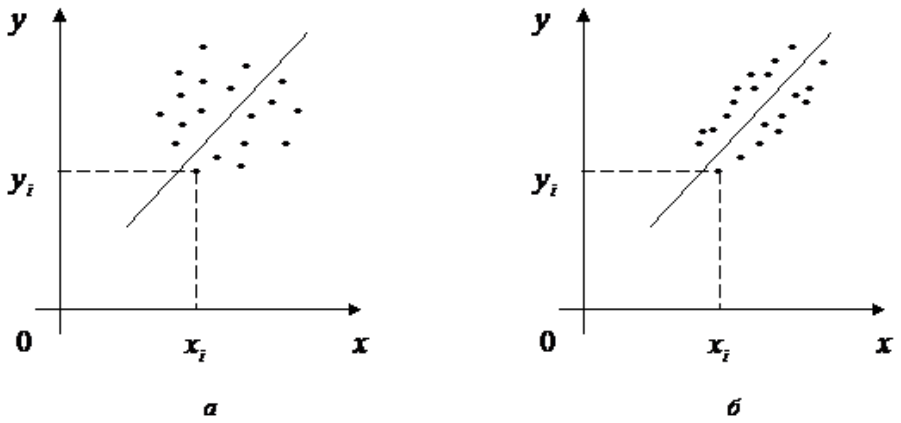


Рис. 1.1 – Кореляційні залежності

2. Якщо $r > 0 (b_1 > 0)$ то кореляційний зв'язок прямий (рис. 1.2.а), якщо $r < 0 (b_1 < 0)$, - обернений (рис. 1.2.б).

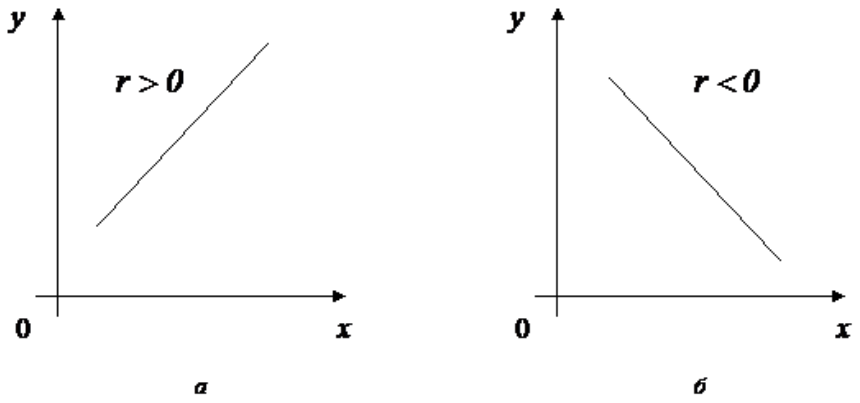


Рис. 1.2 – Прямий та обернений кореляційні зв’язки

3. При $r = \pm 1$ кореляційна залежність являється лінійною функціональною залежністю. При цьому усі значення, що спостерігаються, розташовані на прямій лінії (рис. 1.3).

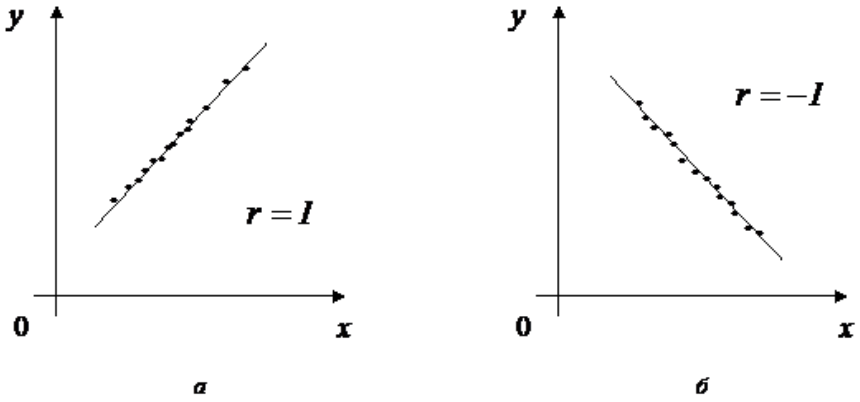


Рис. 1.3. – Лінійна ф-нальна залежність

4. При $r = 0$ лінійна кореляційна залежність відсутня. Це означає або відсутність будь-якої залежності між змінними *x* та *y* (рис. 1.4.а), або належність деякої нелінійної залежності (рис. 1.4.б).

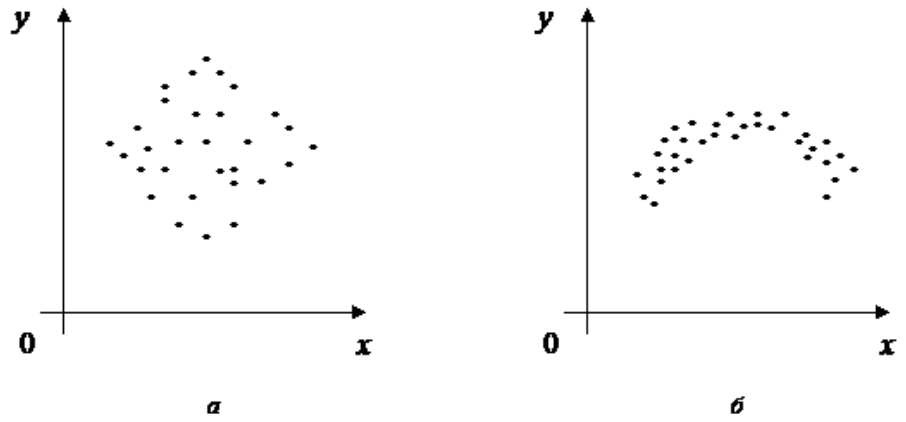


Рис. 1.4. – Відсутність залежності

На практиці для оцінки ступені взаємозв'язку можна керуватись наступними емпіричними правилами:

1) $|r| > 0{,}95$ – існує практично лінійна залежність;

2) $0{,}8 < |r| < 0{,}95$ – сильна ступінь лінійної залежності;

3) $0{,}6 < |r| < 0{,}8$ – належність лінійного зв'язку;

4) $|r| < 0{,}4$ – лінійний зв'язок виявити не вдалося.

Основні припущення регресійного аналізу

Відмітимо основні припущення регресійного аналізу:

1. В моделі (1) похибка ε_i (або залежна зміна y_i) є випадковою величиною, а фактор x_i - невипадкова величина ($i = \overline{1,n}$).

2. Математичне сподівання похибки ε_i дорівнює нулю:

$$M[\varepsilon_i] = 0, i = \overline{1,n}.$$

3. Дисперсія похибки ε_i (або залежної змінної y_i) постійна для будь-якого i:

$$D[\varepsilon_i] = \sigma^2,$$

тобто виконується умова гомоскедастичності (рівнозміненосні похибки).

4. Похибки ε_i та ε_j не корельовані:

$$M[\varepsilon_i, \varepsilon_j] = 0, i \neq j.$$

5. Похибка ε_i (або залежна змінна y_i) являється нормально розподіленою випадковою величиною.

Модель (1), для якої виконуються припущення 1-5 називається класичною нормальною лінійною регрессійною моделлю (CNLR-model).

Для отримання рівняння регресії достатньо припущень 1-4. Вимога виконання припущення 5 (тобто розглядання "нормальної регресії") необхідно для оцінки точності рівняння регресії та її параметрів.

Властивості оцінок параметрів лінійної парної регресії

Оцінкою моделі (1) являється рівняння регресії (2). Параметри цього рівняння визначаються за 1МНК.

Дії неврахованих випадкових факторів та похибок спостережень у моделі (1) визначається за допомогою дисперсії похибок або остаточної дисперсії ε^2. Незміщеною оцінкою цієї дисперсії є вибіркова остаточна дисперсія:

$$S^2 = \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n-2}. \quad (6)$$

Виникає питання, чи являються оцінки a, b, S^2 параметрів α, β, δ^2 "найкращими"?

Оцінка адекватності регресійної моделі. Коефіцієнт детермінації.

Оцінка адекватності регресійної моделі робиться на підставі коефіцієнта детермінації:

$$R^2 = \frac{\sum_{i=1}^{n} \left(\hat{y}_i - \bar{y}\right)^2}{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2}. \quad (10)$$

Величина R^2 показує, яка частка варіації залежної змінної обумовлена варіацією фактора.

Властивості коефіцієнта детермінації:

1. Для ЛПР $R^2=r^2$.

2. Коефіцієнт детермінації приймає значення на відрізку $[0;1]$, тобто $0 \le R^2 \le 1$. Чим ближче R^2 до одиниці, тим краще регресія апроксимує емпіричні дані.

3. Якщо $R^2=1$, між змінними x та y існує лінійна функціональна залежність.

4. Якщо $R^2=0$, то варіація залежної змінної повністю обумовлена впливом випадкових та неврахованих у моделі змінних.

На практиці *для оцінки ступені апроксимації рівнянням регресії вихідних даних* використовують наступні емпіричні правила:

1). $R^2>0,95$ - висока точність апроксимації.

2). $0,8<R^2<0,95$ - задовільна апроксимація.

3). $R^2<0,6$ - незадовільна апроксимація.

Обчислення коефіцієнта еластичності:

Коефіцієнт еластичності E показує - наскільки відсотків (від середньої) змінюється у середньому y при зміненні тільки x на 1% та обчислюється за формулою:

$$E = b_1 \frac{\bar{x}}{\bar{y}}. \qquad (11)$$

4. Довірчі інтервали функції регресії та її параметрів

Довірчий інтервал функції регресії (прогнозу).

Прогнозне значення $\hat{y}_n$ визначається шляхом підстановки в рівняння регресії відповідного значення фактору x_n:

$$\hat{y}_n = \hat{y}(x_n) = b_0 + b_1 x_n. \qquad (12)$$

Довірчий інтервал прогнозу обчислюється за наступними формулами:

$$M_{x_n}[y] = \hat{y}_n \pm t_{1-\alpha;n-2} S_{\hat{y}}, \qquad (13)$$

де $M_{x_n}[y]$ - умовне математичне сподівання залежної змінної; $S_{\hat{y}}$ - оцінка стандартної похибки прогнозу, яка обчислюється за формулою:

$$S_{\hat{y}} = S\sqrt{\frac{1}{n} + \frac{(x_n - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}, \qquad (14)$$

$$S = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n-2}} - \qquad (15)$$

оцінка середньоквадратичного відхилення похибок.

Зауваження. Прогноз значень залежної змінної за рівнянням регресії виправданий, якщо значення *x* пояснюючої змінної *X* не виходить за діапазон її значень за вибіркою.

Довірчі інтервали для коефіцієнтів регресійної моделі.

Формули для обчислення довірчих інтервалів для коефіцієнтів мають наступний вигляд:

$$\beta_0 = b_0 \pm t_{1-\alpha;n-2} \cdot S \cdot \sqrt{\frac{\overline{x^2}}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}; \quad (16)$$

$$\beta_1 = b_1 \pm t_{1-\alpha;n-2} \cdot \frac{S}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}}. \quad (17)$$

Багатофакторна регресія

На практиці економічний процес змінюється під впливом багатьох різноманітних факторів, які треба вміти виявити та оцінити. Якщо розглянути приклад з лк 30 , то аналіз обсягу продажу на фірмі було б спрощено допускати тільки від витрат на рекламу. На обсяги продажу впливає частина ринку, яку утримує фірма, якість продукції, імідж марки продукції, середня заробітна плата населення у регіонах продажу та інші фактори.

Узагальнена багатофакторна лінійна регресійна модель може бути записана у вигляді:

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_px_p + \varepsilon, \quad (1)$$

де у – залежна змінна, x1, x2,…xp – незалежні змінні (фактори) a0, …ap – параметри моделі, які потрібно оцінити, e - не спостережувана випадкова величина.

Узагальнена регресійна модель – це модель, яка дійсна для всієї генеральної сукупності. Невідомі параметри узагальненої моделі є константами, а випадкова величина – не спостережувана, і можна лише зробити припущення відповідно до закону її розподілу. На відміну від

узагальненої регресійної моделі, вибіркова модель будується для певної вибірки; невідомі параметри вибіркової моделі є випадковими величинами, математичне сподівання яких дорівнює параметрам узагальненої моделі.

Відповідна вибіркова лінійна багатофакторна модель має вигляд:

$$y = b_0 + b_1 x_1 + \ldots + b_p x_p + e, \qquad (2)$$

де ŷ – залежна змінна, x1…xp – незалежні змінні, b0, b1…bp – оцінки невідомих параметрів узагальненої моделі, e – випадкова величина (помилка).

Нехай дано ряд спостережень за залежною змінною $y = \{y_1, \ldots y_n\}$ та за незалежними змінними, або факторами: $x_1 = (x_{11}, x_{12}, \ldots x_{1n}), \ldots x_p = (x_{p1}, x_{p2}, \ldots x_{pn})$. На підставі цих спостережень будується лінійна вибіркова багатофакторна модель, а саме – у вигляді (2).

Як і у випадку простої лінійної регресії, знаходять невідомі параметри за методом найменших квадратів, тобто мінімізують суму квадратів відхилень фактичних даних від теоретичних: $F(b, b_1, \ldots b_n) = \min \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_{1i} - \ldots b_p x_{pi})^2$, тобто $\frac{\partial F}{\partial b_i} = 0 \left(i = \overline{0,8}\right)$.

Звідки отримується нормальна система рівнянь:

$$\begin{cases} nb_0 + b_1 \sum_{i=1}^{n} x_{1i} + \ldots b_p \sum_{i=1}^{n} x_{pi} = \sum_{i=1}^{n} y_i \\ b_0 \sum_{i=1}^{n} x_{1i} + b_1 \sum_{i=1}^{n} x_{1i}^2 + \ldots b_p \sum_{i=1}^{n} x_{pi} x_{1i} = \sum_{i=1}^{n} y_i x_{1i} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ b_0 \sum_{i=1}^{n} x_{pi} + b_1 \sum_{i=1}^{n} x_{1i} x_{pi} + \ldots b_p \sum_{i=1}^{n} x_{pi}^2 = \sum_{i=1}^{n} y_i x_{pi} \end{cases} \qquad (3)$$

Розв’язуючи систему рівнянь (3) щодо b0, b1…bp одержують рівняння множинної регресії.

Лінійну багатофакторну модель, як і основні проблеми регресійного аналізу, зручно розглядати за допомогою матриць. Для цього введемо матриці:

$$(n\times 1)=\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},\ B((p+1)\times 1)=\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{pmatrix},\ X(n\times(p+1))=\begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},\ E(n\times 1)=\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

Тоді систему (2) можна записати у матричній формі

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{pmatrix}=\begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}+\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix},\quad y=\hat{XB}+E$$

, (4)

а систему (3) можна записати у такому матричному вигляді:

$$\begin{pmatrix} n\sum_{i=1}^{n}x_{1i} & \sum_{i=1}^{n}x_{2i} & \cdots & \sum_{i=1}^{n}x_{pi} \\ \sum_{i=1}^{n}x_{1i} & \sum_{i=1}^{n}x_{1i}^2 & \cdots & \sum x_{1i}x_{pi} \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{i=1}^{n}x_{pi} & \sum_{i=1}^{n}x_{pi}x_{1i} & \cdots & \sum_{i=1}^{n}x_{pi}^2 \end{pmatrix}\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{pmatrix}=\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_{11} & x_{12} & \cdots & x_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{p1} & x_{p2} & \cdots & x_{pn} \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

, (5)

або XTXB =XTY Якщо обернена матриця XTX існує (XTX)-1, то, помноживши на неї останню рівність, отримаємо: (XTX)-1(XTX) B = (XTX)-1 XTY, або остаточно:

$$B=\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{pmatrix}=\left(X^TX\right)^TX^TY$$

(6)

Рівняння (6) є фундаментальним результатом для визначення невідомих вибіркових параметрів у матричному вигляді.

Нелінійна регресія

Якщо графік регресії $\bar{y}_x=f(x)$ або $\bar{x}_y=\varphi(y)$ зображується кривою лінією, то кореляцію називають криволінійною (нелінійною). Наприклад, квадратичні

функції використовуються для опису дуже широкого спектру економічних процесів, завдяки їхнім універсальним властивостям. Дійсно, у загальному випадку квадратична функція має вигляд: $y = ax^2 + bx + c$. (1)

Обернена функція має вигляд: $y = \frac{a}{x} + b$, (2)

тобто узагальнені регресії моделі відповідно будуть:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$$
$$y = \beta_0 + \frac{\beta_1}{x} + \varepsilon \quad , \qquad (3)$$

а вибіркові нелінійні регресії є:

$$\hat{y} = b_0 + b_1 x + b_2 x^2 + e \qquad (4)$$

$$\hat{y} = b_0 + \frac{b_1}{x} + e \qquad (5)$$

За методом найменших квадратів параметри b_0, b_1, b_2 знаходять з системи:

$$\begin{cases} b_2 \sum_{i=1}^{n} x_i^2 + b_1 \sum_{i=1}^{n} x_i + b_0 n = \sum_{i=1}^{n} y_i \\ b_2 \sum_{i=1}^{n} x_i^3 + b_1 \sum_{i=1}^{n} x_i^2 + b_0 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \\ b_3 \sum_{i=1}^{n} x_i^4 + b_1 \sum_{i=1}^{n} x_i^3 + b_0 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i^2 y_i \end{cases} , \qquad (6)$$

$$\begin{cases} b_1 \sum_{i=1}^{n} \frac{1}{x_i} + b_0 n = \sum_{i=1}^{n} y_i \\ b_1 \sum_{i=1}^{n} \frac{1}{x_i^2} + b_0 \sum_{i=1}^{\bar{n}} \frac{1}{x_i} = \sum_{i=1}^{n} \frac{y_i}{x_i} \end{cases} \qquad (7)$$

Для аналізу зв’язку у і х в цих випадках використовуються кореляційними відношеннями:

$$\beta_x = \sqrt{\frac{\sum_{i=1}^{n} \left(\hat{y}_i - \bar{y} \right)^2}{\sum_{i=1}^{n} \left(y_i - \bar{y} \right)^2}} \quad , \qquad (8)$$

де $0 \le \beta_x \le 1$.

1.3 Порядок виконання роботи

1.3.1 Лінійна регресія

Створимо новий файл, в якому змінну *VAR1* заповнимо послідовно значеннями від 0 до 10, змінну *VAR2* – випадковими значеннями від 0 до 1, а змінну *VAR*3 задамо, як суму *VAR1+ VAR2*.

Виконаємо послідовність команд: *Statistics -> Basic Statistics/Tables -> Descriptive Statistics -> Prob.&Scatterplots* (див. рис. 1.3.1).

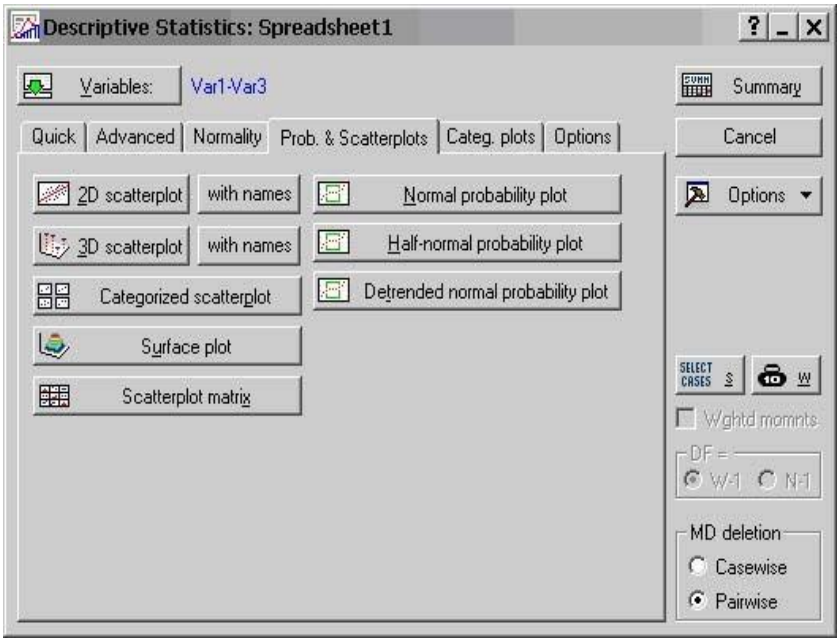


Рис. 1.3.1 – Виконання команд

Як Variables виберемо *VAR1-VAR3,* натиснемо кнопку *2D Scatterplot.*

У першому списку змінних вкажемо *VAR1*, в другому – *VAR3* і натиснемо кнопку *OK* (див. рис. 1.3.2).

На графіку, що з’явився, зображено пряму лінійної регресійної моделі для *VAR3* через *VAR1,* а у верхній частині вікна бачимо рівняння лінійної регресії (див. рис. 1.3.3).

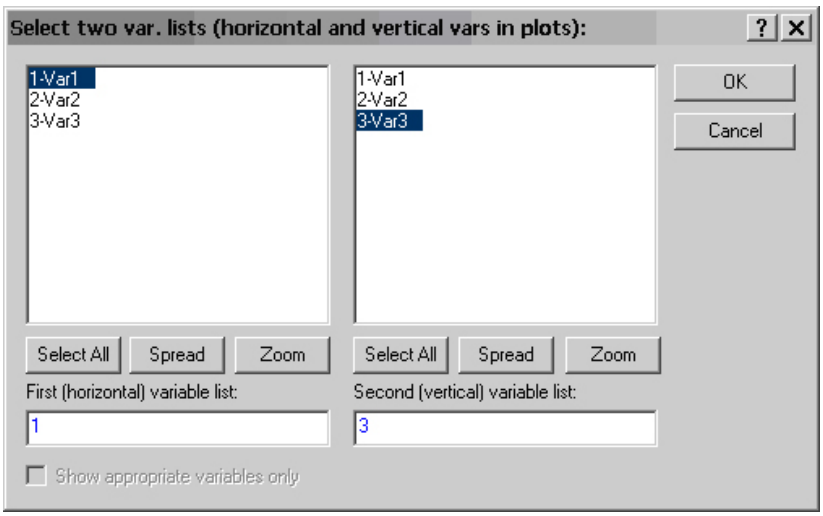


Рис. 1.3.2 – Вибір параметрів

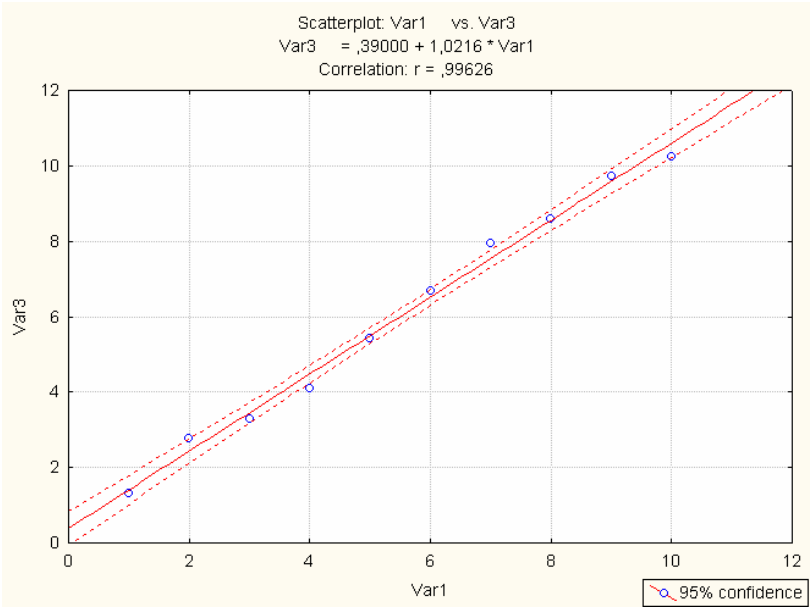


Рис. 1.3.3 – Отримана пряма лінійної регресії

Якщо у змінній *VAR3* замінити одне із значень, наприклад на 70, і побудувати графік знову, то побачимо, що рівняння регресії буде враховувати дане значення і з графіка буде очевидно, що 70 є викидом (див. рис. 1.3.4).

Натиснемо піктограму *Brushing*. У вікні, що з'явиться зробимо активними *Exclude* та *Box*, виділимо прямокутником значення (див. рис. 1.3.5) і натиснемо кнопку *Apply*. Виділене значення зникне з графіка і регресійна пряма змінить своє положення.

Для того, щоб значення викиду не виводилось у наступних графіках та не

враховувалось при обчисленні регресійної формули, у змінній *VAR4* заповнимо всі значення одиницями, а те значення, що стоїть напроти викиду – нулем (див. рис. 1.3.6).

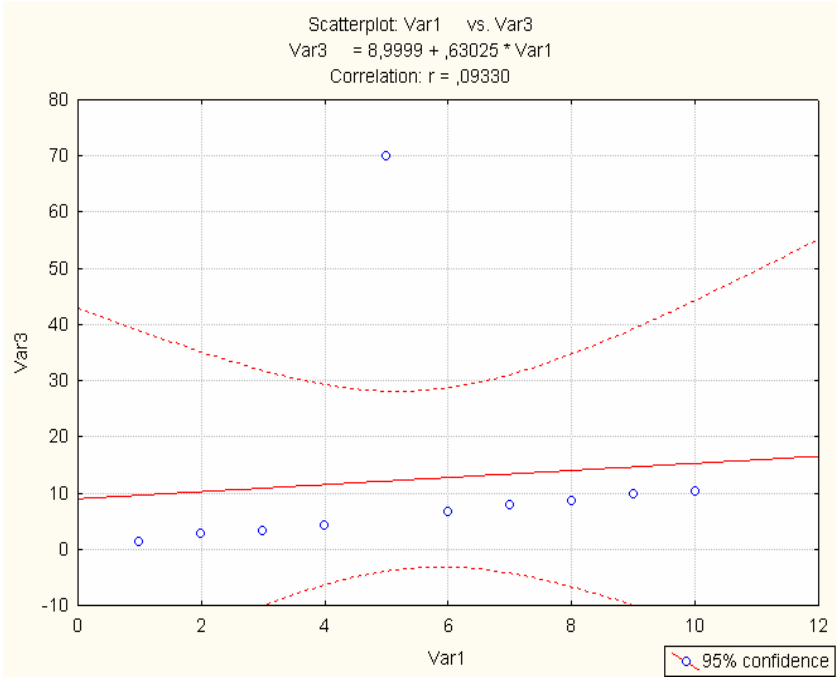


Рис. 1.3.4 – Заміна значення та отримання викиду

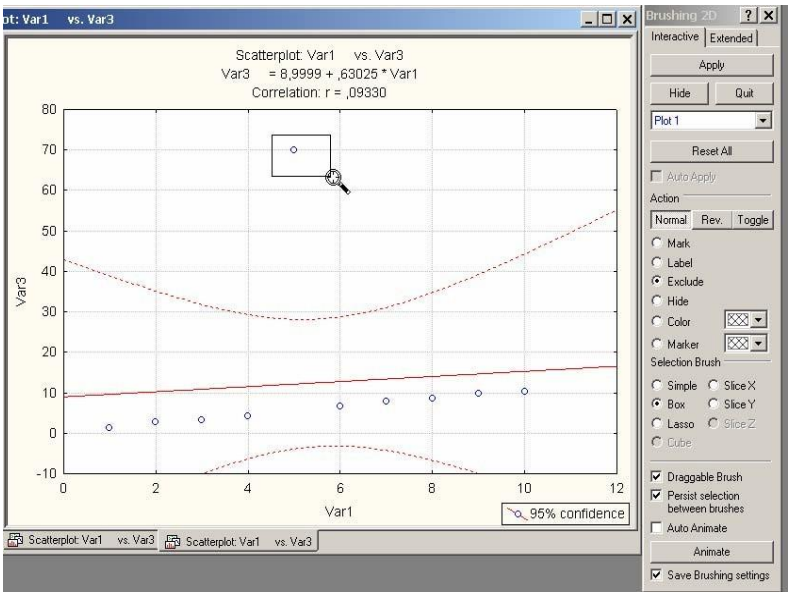


Рис. 1.3.5 – Зміна положення прямої

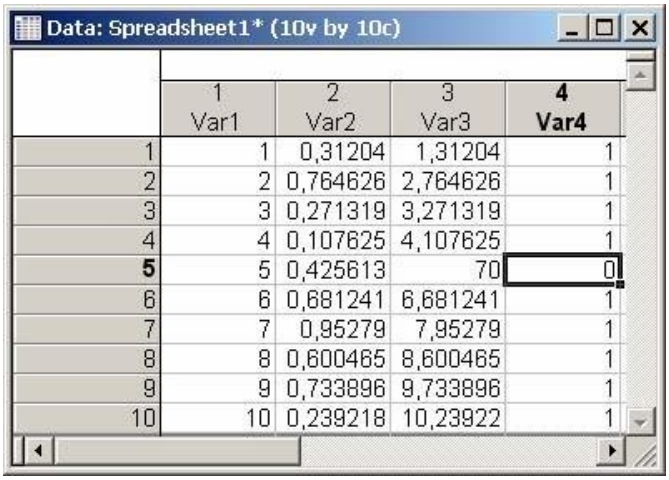


Рис. 1.3.6 – Прибираємо викид

У вікні *Descriptive Statistics* натиснемо кнопку *Weight* (див. рис. 1.3.5). Оберемо змінну *VAR4*, перемкнемо *Status* на *On* та натиснемо *OK* (див. рис. 1.3.7). Тепер при аналізі викиди враховуватися не будуть.

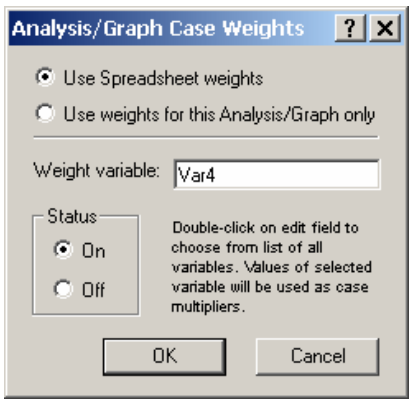


Рис. 1.3.7 – Вікно Graph Case Weight

Нехай, маємо таку таблицю з даними:

Y	*X*	*Z*
40	100	10
50	200	20
50	300	10
70	400	30
65	500	20
65	600	20
80	700	30

де *Y* – врожайність, *X* – добрива, *Z* – опади.

Потрібно знайти формулу багатофакторної лінійної регресії для *Y*:

$$Y=B_0 +B_1 X+B_2 Z,$$

де B_i – невідомі коефіцієнти.

Виконаємо послідовність команд: *Statistics -> Multiple Regression*, як змінні оберемо Y – залежна, X і Z – незалежні (див. рис. 1.3.8). Натиснемо *OK*. Отримуємо результат, який зображено на рис. 1.3.9.

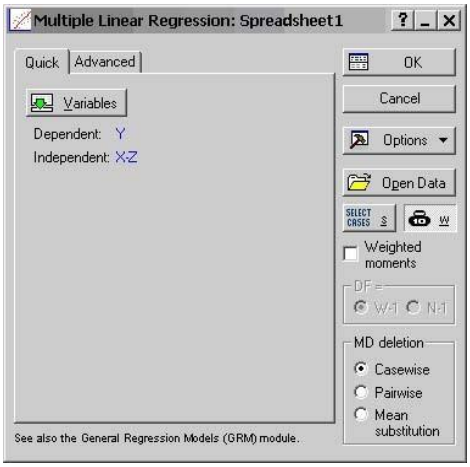


Рис. 1.3.8 – Вікно багатофакторної регресії

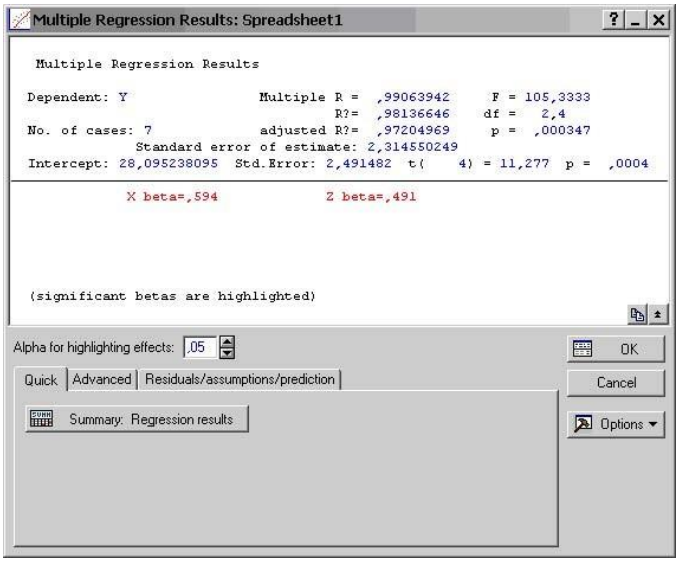


Рис. 1.3.9 – Результат багатофакторної регресії

В закладці *Quick* натиснемо кнопку *Summary: Regression results.*У вікні, що з’явилось (див. рис. 1.3.10), бачимо оцінки параметрів та допоміжну статистику. Обидві змінні значимі (виділені червоним). У третьому стовпці таблиці вказані оцінки для коефіцієнтів B_i . Отже, *Y=28.095+0.038*X+0.833*Z.*

Якщо наші змінні попередньо стандартизувати, то у результаті такого регресійного аналізу отримали б оцінки коефіцієнтів, які записані в першому стовпчику таблиці (зрозуміло, що $B_0=0$). Коефіцієнти з першого стовпця показують внесок у регресійну модель змінних *X* та *Z*.

Regression Summary for Dependent Variable: Y (Spreadsheet1)
R= ,99063942 R?= ,98136646 Adjusted R?= ,97204969
F(2,4)=105,33 p<,00035 Std.Error of estimate: 2,3146

N=7	Beta	Std.Err. of Beta	B	Std.Err. of B	t(4)	p-level
Intercept			28,09524	2,491482	11,27652	0,000352
X	0,594430	0,091003	0,03810	0,005832	6,53197	0,002838
Z	0,491473	0,091003	0,83333	0,154303	5,40062	0,005690

Рис. 1.3.10 – Оцінки параметрів та допоміжна статистика

Для того, щоб обчислити передбачуване значення для *Y* для заданих *X* та *Z* і побудувати 95% проміжок надійності, перейдемо у закладку *Residuals/assumptions/prediction* і натиснемо *Predict dependent variable* (див. рис. 1.3.11). У відповідні віконця вводимо значення змінних *X* та *Z* (див. рис. 1.3.12) і натискаємо *OK*.

Якщо незалежні змінні набувають одного і того ж значення його можна ввести у віконці *Common Value* і натиснути *Apply*.

У вікні результатів аналізу (див. рис. 1.3.13) бачимо передбачуване значення *Y*, верхню і нижню межу надійного проміжку для цього значення.

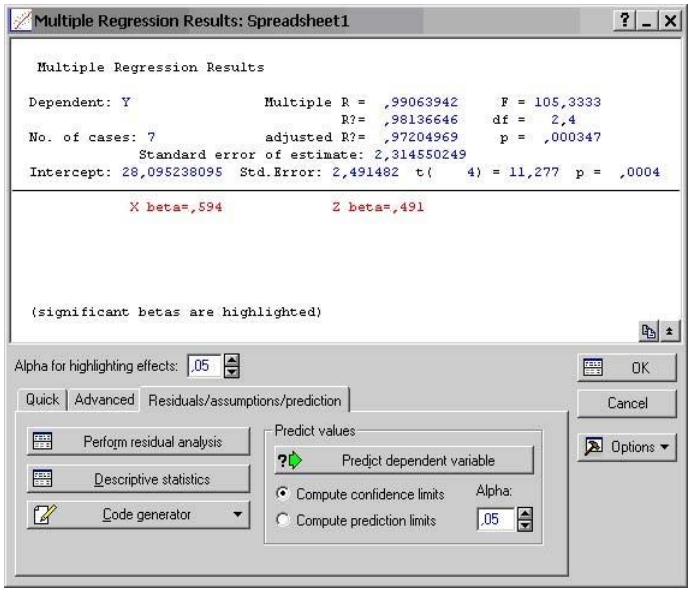


Рис. 1.3.11 – Закладка *Residuals/assumptions/prediction*

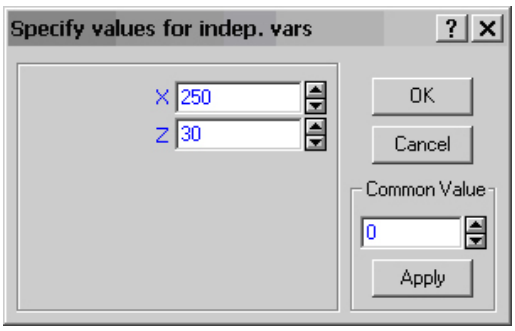


Рис. 1.3.12 – Введення значень змінних

Variable	Predicting Values for (Spreadsheet1) variable: Y **B-Weight**	Value	B-Weight * Value
X	0,038095	250,0000	9,52381
Z	0,833333	30,0000	25,00000
Intercept			28,09524
Predicted			62,61905
-95,0%CL			55,99196
+95,0%CL			69,24614

Рис. 1.3.13 – Результати аналізу

Якщо ж потрібно подивитись як розподілені залишки, то натиснемо кнопку *Perform residual analysis*. У вікні, що з'явилось (див. рис. 1.3.14), зібрані різні методи для аналізу залишків регресійної моделі.

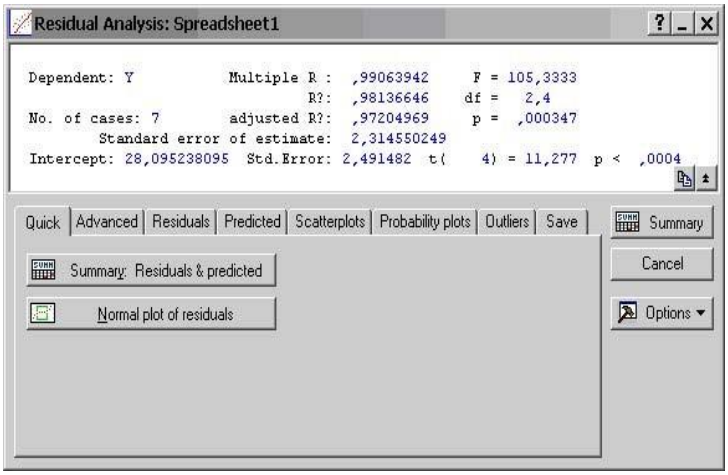


Рис. 1.3.14 – Методи для аналізу залишків регресійної моделі

Наприклад, натиснувши *Normal plot of residuals* отримаємо *Q-Q* графік, на якому видно наскільки залишки узгоджуються з нормальним законом розподілу (див. рис. 1.3.15).

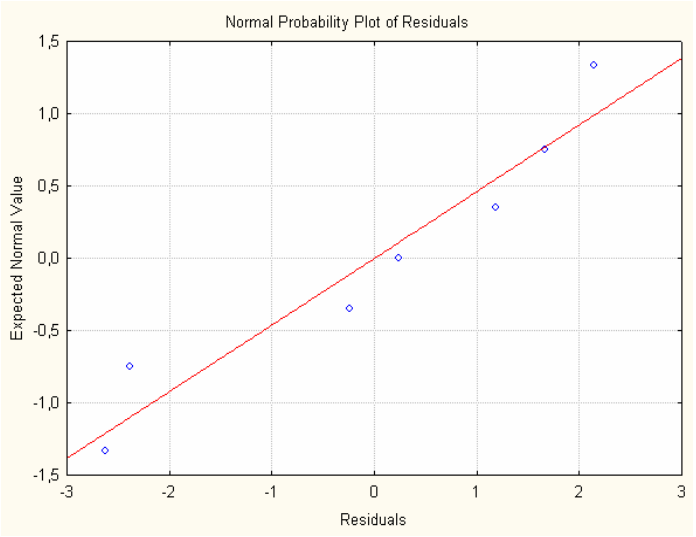


Рис. 1.3.15 – *Normal plot of residuals*

1.3.2 Багатофакторна регресія

Відкриємо файл *Job_prof.sta* (див. рис. 1.3.1). У файлі вказано бали, отримані претендентами під час тестування при прийнятті на посаду (у перших чотирьох стовпцях), та оцінка професійної здатності претендентів (у п'ятому стовпці) після закінчення випробувального терміна. Нам потрібно знайти лінійну багатофакторну регресійну модель залежності оцінки професійної здатності від оцінок за тести. Завантажимо модуль *Multiple Regression: Statistics -> Multiple Regression*.

Job proficiency data set from Neter, Was:

	1 TEST1	2 TEST2	3 TEST3	4 TEST4	5 JOB_PROF
1	86	110	100	87	88
2	62	97	99	100	80
3	110	107	103	103	96
4	101	117	93	95	76
5	100	101	95	88	80
6	78	85	95	84	73
7	120	77	80	74	58
8	105	122	116	102	116
9	112	119	106	105	104
10	120	89	105	97	99
11	87	81	90	88	64
12	133	120	113	108	126
13	140	121	96	89	94
14	84	113	98	78	71
15	106	102	109	109	111
16	109	129	102	108	109
17	104	83	100	102	100
18	150	118	107	110	127
19	98	125	108	95	99
20	120	94	95	90	82
21	74	121	91	85	67
22	96	114	114	103	109
23	104	73	93	80	78
24	94	121	115	104	115
25	91	129	97	83	83

Рис. 1.3.1 – Відкритий файл

Натиснувши *Variables,* обираємо *Job-Prof,* як залежну змінну, а як незалежні змінні вибираємо перші чотири змінні (див. рис. 1.3.2). Двічі натискаємо *OK.* Результати регресійного аналізу зображені на рисунку 1.3.3. Всі змінні, окрім другої, є значимими (виділені червоним).

У закладці *Quick* натиснемо *Summary: Regression result* (див. рис. 1.3.3). У вікні, що з'явилось (див. рис. 1.3.4), бачимо результати аналізу: у третьому стовпці – коефіцієнти багатофакторної лінійної регресійної моделі, а в першому стовпці – коефіцієнти цієї ж регресійної моделі для стандартизованих змінних.

Проаналізувавши результати прийдемо до висновку, що *Test 2* досить мало впливає на оцінку професійної здатності претендентів: відповідний коефіцієнт у першому стовбці становить 0,043. Тому можливо є доречним взагалі вилучити *Test 2* з регресійної моделі.

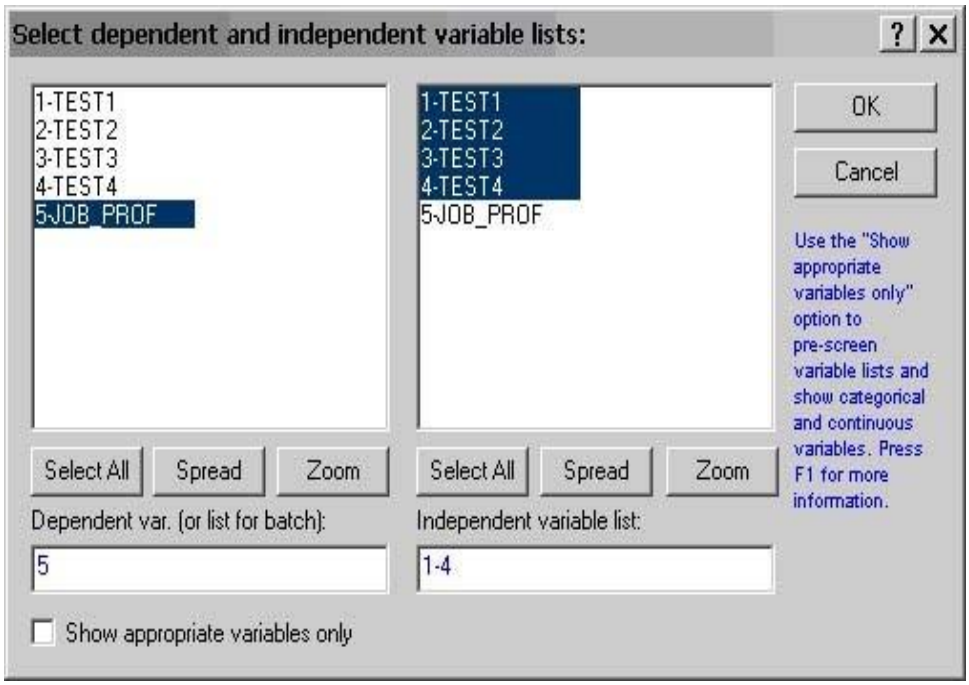


Рис. 1.3.2 – Налаштовуємо змінні

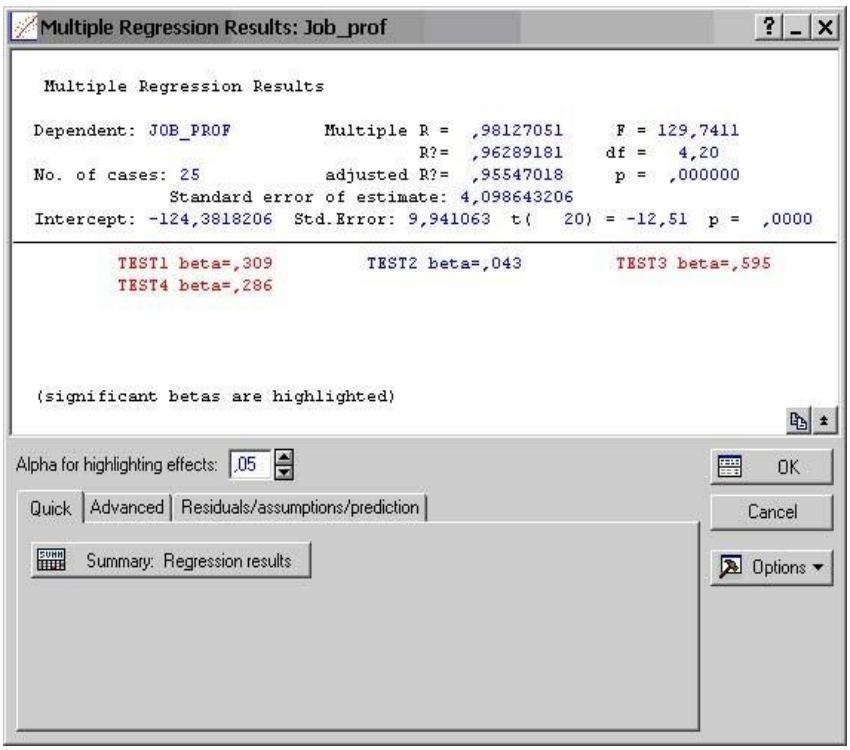


Рис. 1.3.3 – Результати регресійного аналізу

Regression Summary for Dependent Variable: JOB_PROF (Jo
R= ,98127051 R?= ,96289181 Adjusted R?= ,95547018
F(4,20)=129,74 p<,00000 Std.Error of estimate: 4,0986

N=25	Beta	Std.Err. of Beta	B	Std.Err. of B	t(20)	p-level
Intercept			-124,382	9,941063	-12,5119	0,000000
TEST1	0,309042	0,045951	0,296	0,043971	6,7254	0,000002
TEST2	0,042992	0,050408	0,048	0,056617	0,8529	0,403826
TEST3	0,595438	0,074813	1,306	0,164091	7,9591	0,000000
TEST4	0,285723	0,072524	0,520	0,131943	3,9397	0,000810

Рис. 1.3.4 – Результати регресійного аналізу

Проведемо спочатку аналіз залишків побудованої моделі. У закладці *Residuals/assumptions/prediction*, натиснемо кнопку *Perform residual analysis*. У вікні, що з'явилось, обираємо *Residuals -> Casewise plot of residuals* (див. рис. 1.3.5).

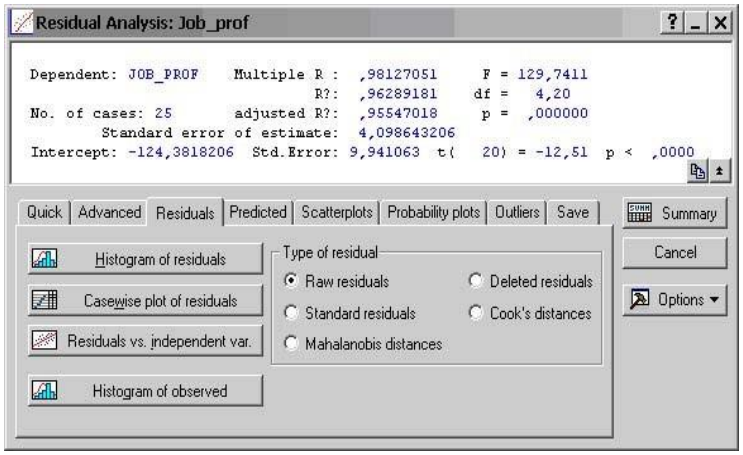


Рис. 1.3.5 – Аналіз залишків

Так ми перевіримо, чи не виходять залишки за межі 3δ. Отримаємо таблицю (див. рис. 1.3.6), в якій знаком «*» вказано де знаходиться залишок у інтервалі (-3δ; 3δ). У правій частині таблиці міститься додаткова інформація про залишки.

Бачимо, що залишки лежать у проміжку (-2δ; 2δ) і їхні середнє і медіана дорівнюють нулю.

Потім оберемо закладку *Outliers* (див. рис. 1.3.7). Натиснемо *Casewise plot outliers* та упевнимося, що викидів немає – з’явиться відповідне повідомлення (див. рис. 1.3.8).

Raw Residual (Job_prof)
Dependent variable: JOB_PROF

Case	Raw Residuals -3σ … 0 … +3σ	Observed Value	Predicted Value	Residual	Standard Pred. v	Standard Residual	Std.Err. Pred.Val	Mahalanobis Distance	Deleted Residual	Cook's Distance
1		88,0000	82,1876	5,81243	-0,52533	1,41814	1,307494	1,402366	8,47095	0,050732
2		80,0000	79,8141	0,08584	-0,64462	0,02087	2,428804	7,475472	0,13251	0,000073
3		96,0000	101,3753	-5,37526	0,48141	-1,31147	1,158157	0,956312	-5,84170	0,032440
4		76,0000	81,9779	-5,97794	-0,53633	-1,45852	1,022519	3,705427	-7,45125	0,130899
5		80,0000	79,8829	0,11710	-0,64626	0,02857	0,987756	0,462262	0,12447	0,000011
6		73,0000	70,5251	2,47494	-1,13725	0,60384	1,686933	3,105818	2,97970	0,017906
7		58,0000	57,7709	0,22913	-1,80644	0,05590	2,478193	7,814074	0,36117	0,000569
8		116,0000	117,0783	-1,07827	1,30537	-0,26332	1,875513	4,065404	-1,36511	0,004646
9		104,0000	107,5038	-3,50383	0,80297	-0,85488	1,244252	1,251808	-3,85952	0,016344
10		99,0000	102,9564	-3,95644	0,58437	-0,96530	1,725421	3,293252	-4,80081	0,048706
11		64,0000	68,5427	-4,54266	-1,24126	-1,10833	1,736381	3,347456	-5,53630	0,065494
12		126,0000	124,4639	1,53612	1,69283	0,37479	1,752489	3,427749	1,87979	0,007691
13		94,0000	94,5035	-0,50350	0,12088	-0,12295	2,247540	6,258816	-0,72001	0,001858
14		71,0000	74,4506	-3,45059	-0,93128	-0,84189	2,038646	4,877651	-4,58491	0,061918
15		111,0000	110,9059	0,09410	0,98147	0,02296	1,600871	2,701370	0,11104	0,000022
16		109,0000	103,4349	5,56508	0,58940	1,35779	2,144107	5,808351	7,66200	0,191284
17		100,0000	94,0041	5,99586	0,09466	1,46288	1,847200	3,814823	7,52414	0,136803
18		127,0000	122,5982	4,40179	1,59494	1,07396	2,223710	6,104595	6,23799	0,136369
19		99,0000	101,0872	-2,08723	0,46525	-0,50437	1,476318	2,153791	-2,37542	0,009718
20		82,0000	86,4880	-4,48803	-0,29812	-1,09768	1,337886	1,587607	-5,03567	0,032173
21		67,0000	66,3763	0,62370	-1,35492	0,15217	2,157740	5,891650	0,86284	0,002457
22		109,0000	111,9392	-2,93924	1,03580	-0,71713	1,688429	3,007389	-3,52135	0,024404
23		78,0000	72,9432	5,05683	-1,01037	1,23378	2,117101	5,443451	6,88703	0,151104
24		115,0000	113,5116	1,48837	1,11818	0,36314	1,729615	3,313956	1,81085	0,006952
25		83,0000	78,5884	4,41384	-0,71429	1,07695	2,001189	4,761352	5,79515	0,095316
Minimum		58,0000	57,7709	-5,97794	-1,80644	-1,45852	0,987756	0,462262	-7,45125	0,000011
Maximum		127,0000	124,4639	5,99586	1,69283	1,46289	2,478193	7,814074	7,66200	0,191284
Mean		92,2000	92,2000	0,00000	0,00000	0,00000	1,792141	3,840000	0,14999	0,046995
Median		94,0000	94,0041	0,09410	0,09466	0,02296	1,752489	3,427749	0,12447	0,024404

Рис. 1.3.6 – Таблиця залишків

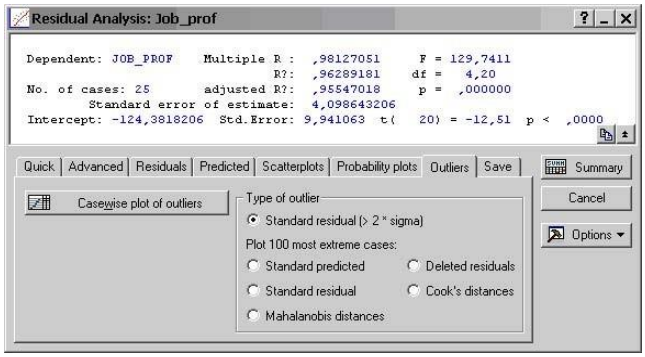


Рис. 1.3.7 – Закладка Outliers

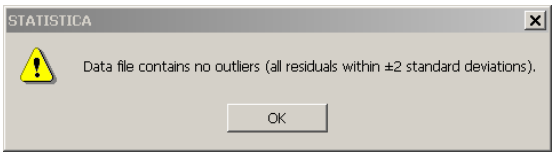


Рис. 1.3.8 – Отримане повідомлення

На основі цих результатів можна вважати, що багатофакторна регресійна модель достатньо добре описує наші дані.

Цікаво дослідити, що відбудеться, якщо вилучити змінну *Test 2,* яка досить мало впливає на оцінку професійної здатності претендентів, з регресійної моделі. Розглянемо, як автоматизовано процес знаходження змінних, які дають малий внесок у регресійну модель, у пакеті *Statistica.*

У вікні *Multiple Regression* у закладці *Advanced* відмітимо *Advanced options (stepwise or ridge regression)* та натиснемо *OK* (див. рис. 1.3.9).

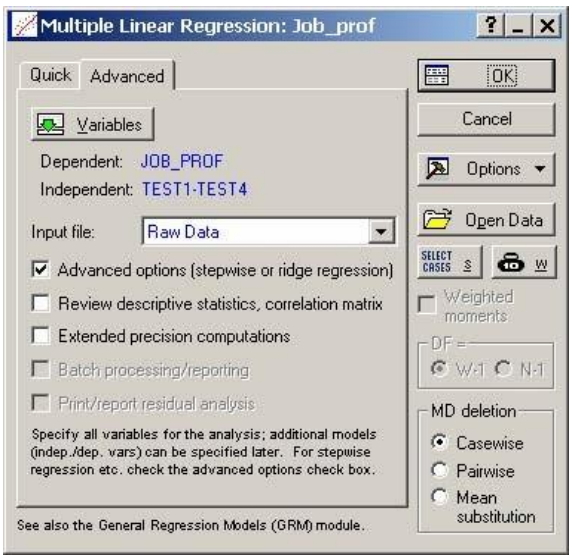


Рис. 1.3.9 – Налаштування

У закладці *Stepwise* вкажемо метод *Forward stepwise* – змінні будуть введені у регресійну модель по одній. У *Display results* вкажемо покроковий вивід результатів – *At each step* (див. рис. 1.3.10). Тобто ми будемо здійснювати покрокову регресію, з виводом результатів після кожного кроку. У полі *F to enter* вказуємо значення 1, а у полі *F to remove* вказуємо 0.01. Ці два числа визначають верхню та нижню межі проміжку для значимості внеску у регресійну модель змінних. Якщо значимість змінної потрапляє в цей проміжок, то включаємо її до регресійної моделі, інакше – відкидаємо.

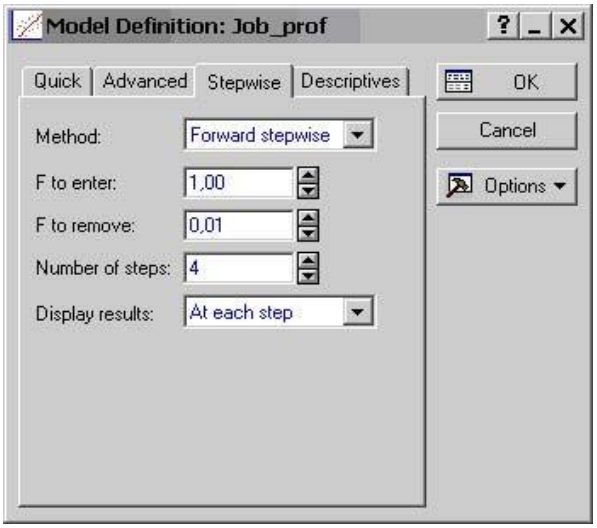


Рис. 1.3.10 – Покроковий вивід результату

Оскільки маємо чотири незалежні змінні, то кількість кроків множинної регресії *Number of steps* досить вказати рівною чотирьом (після кожного кроку до моделі може включатись не більше однієї змінної). Натискаємо кнопку *OK*.

У вікні, що з'явилось, бачимо, що в моделі ще немає жодної змінної (див. рис. 1.3.11). Натискаємо *Next*. З'явився перший коефіцієнт та перша, вибрана до рівняння регресії змінна (див. рис. 1.3.12). Натискаємо кнопку *Next*, допоки на цій кнопці не з'явиться напис *OK*. Це означатиме, що процедуру вибору змінних до регресійної моделі закінчено і всі змінні значимість внеску яких знаходиться у вказаних межах увійшли до рівняння регресії. Бачимо, що змінна *Test 2* не увійшла у нову багатофакторну регресійну модель (див. рис. 1.3.13).

Далі можемо подивитися результати. Для цього натиснемо на закладці *Advanced*, кнопку *Stepwise Regression Summary* (див. рис. 1.3.14). У вікні, що з'явилось (див. рис. 1.3.15), бачимо, статистику внеску обраних змінних і порядок включення їх у регресійну модель. Якщо натиснемо на закладці *Advanced*, кнопку *Summary: Regression results* (див. рис. 1.3.14), то у вікні, що з'явилось (див. рис. 1.3.16), побачимо з якими коефіцієнтами змінні увійшли до регресійної моделі.

Використовуючи отриману інформацію (рис.1.3.15 і 1.3.16) можемо порівняти стару і нову регресійну моделі. Бачимо, що вони майже не відрізняються. Отже, внесок змінної *Test 2* дійсно був незначним.

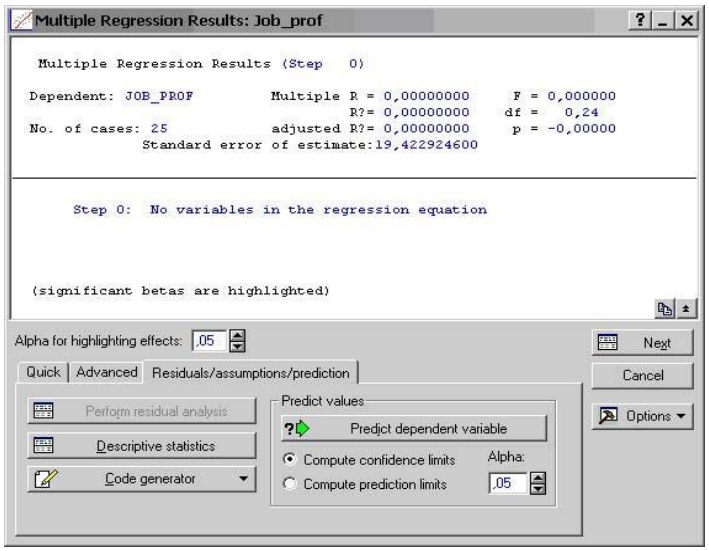


Рис. 1.3.11 – Перевірка моделі перший крок

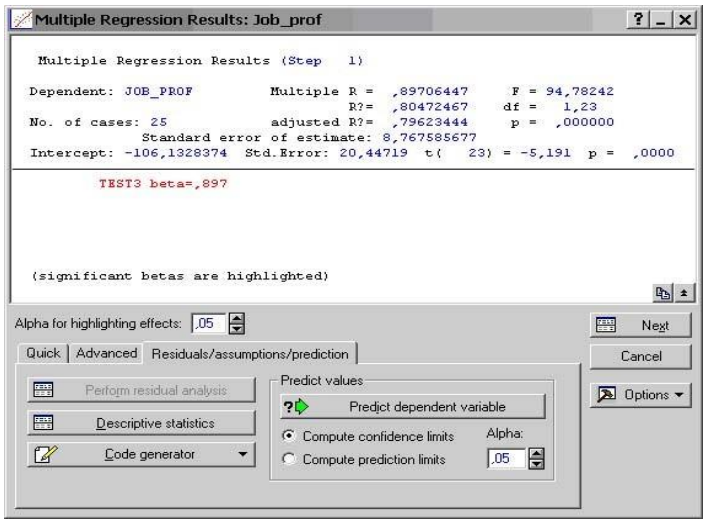


Рис. 1.3.12 – Другий крок

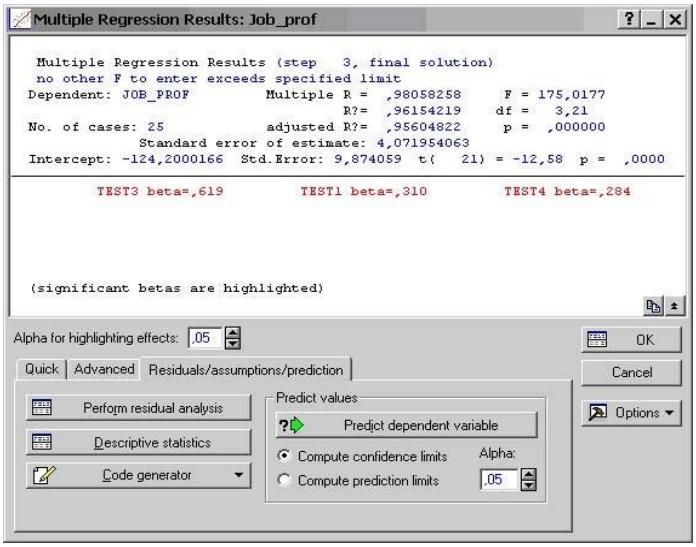


Рис. 1.3.13 – Перевірка моделі насупні кроки

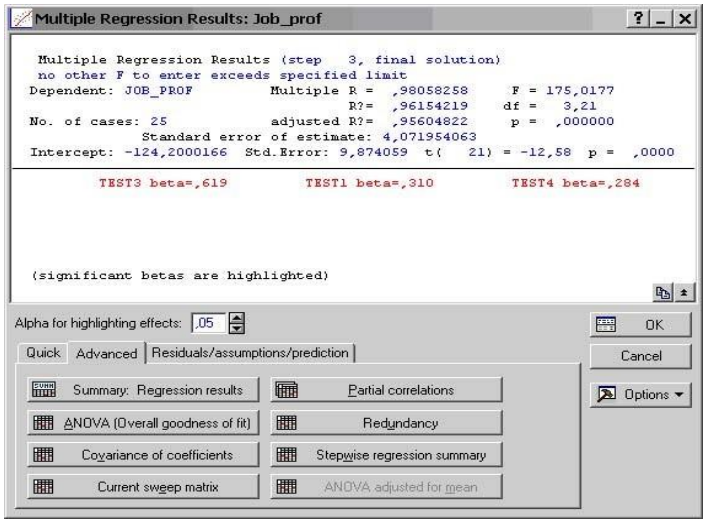


Рис. 1.3.14 – Вкладка Advanced

Summary of Stepwise Regression; DV: JOB_PROF (Job_prof)

Variable	Step +in/-out	Multiple R	Multiple R-square	R-square change	F - to entr/rem	p-level	Variables included
TEST3	1	0,897064	0,804725	0,804725	94,78241	0,000000	1
TEST1	2	0,965917	0,932996	0,128271	42,11609	0,000002	2
TEST4	3	0,980583	0,961542	0,028547	15,58793	0,000735	3

Рис. 1.3.15 – Статистика внеску обраних змінних

Regression Summary for Dependent Variable: JOB_PROF (Jo
R= ,98058258 R?= ,96154219 Adjusted R?= ,95604822
F(3,21)=175,02 p<,00000 Std.Error of estimate: 4,0720

N=25	Beta	Std.Err. of Beta	B	Std.Err. of B	t(21)	p-level
Intercept			-124,200	9,874059	-12,5784	0,000000
TEST3	0,618670	0,069224	1,357	0,151832	8,9373	0,000000
TEST1	0,309670	0,045646	0,296	0,043679	6,7841	0,000001
TEST4	0,284405	0,072035	0,517	0,131054	3,9482	0,000735

Рис. 1.3.16 – Коефіцієни змінних, що увійшли до регресійної моделі

Якщо є мультиколінеарність змінних (наприклад, в кількох стовпцях містяться дані про ціну одного і того ж товару в різних валютах), то очевидно, що для регресійної моделі слід взяти не всі із таких змінних.

Якщо для обробки таблиці мультиколінеарних даних так як і раніше скористатися стандартним алгоритмом, то з'явиться повідомлення про помилку (див. рис. 1.3.17).

Рис. 1.3.17 – Повідомлення про помилку

У цьому разі для регресійного аналізу потрібно вибирати метод *Forward stepwise* (який дозволяє вводити у модель по одній змінній), а не *Standard*.

1.3.3 Нелінійна регресія

Досить часто із діаграми розсіювання, або попереднього досвіду роботи з подібними даними зрозуміло, що регресійна модель не є лінійною. Отже, виникає ситуація, коли потрібно розглядати регресійну модель, у яку входять не просто змінні, а деякі функції від них (наприклад, степінь, експонента, логарифм). Покажемо, як діяти у цьому випадку.

Створимо новий файл. Першу змінну заповнимо числами від 1 до 10, другу – випадковими значеннями від 0 до 1, третя змінна *v3=2*v1*v1+v2+20*.

Потрібно отримати рівняння регресії для *VAR3* через *VAR1*. Зрозуміло, що доцільно ввести у регресійну модель квадрат *VAR1*.

Виконаємо *Statistics -> Advanced Linear/Nonlinear Models -> Fixed nonlinear Regression* (див. рис. 1.3.1).

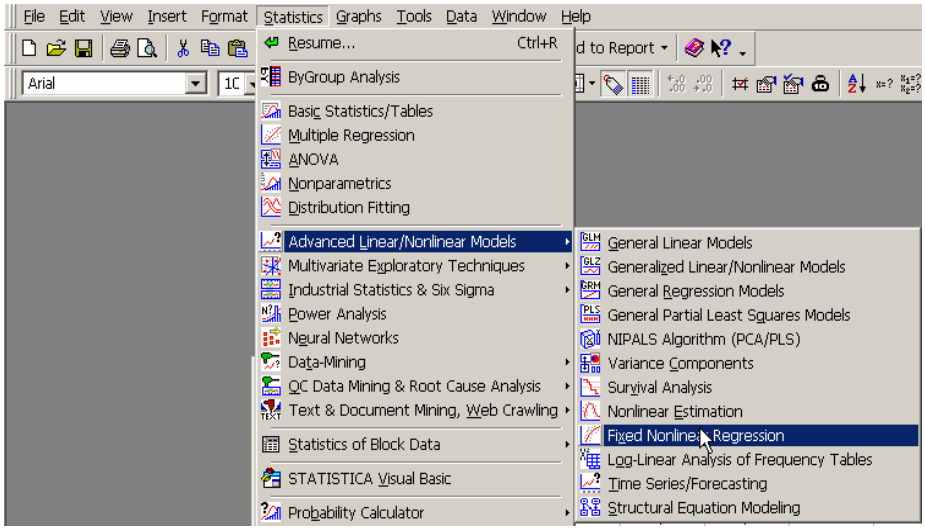


Рис. 1.3.1 – Обираємо нелінійну регресію

У вікні, що з'явиться (див. рис. 13.2), оберемо зміні *VAR1* та *VAR3*. Натискаємо *OK*. Оскільки ми знаємо, що наша залежність квадратична, то виберемо *X**2 (див. рис. 1.3.3). Натискаємо *OK*.

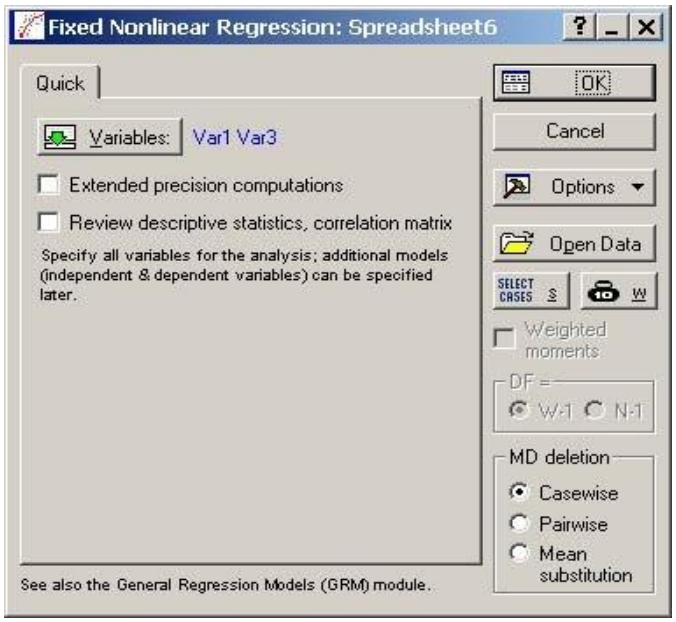


Рис. 1.3.2 – Обираємо змінні

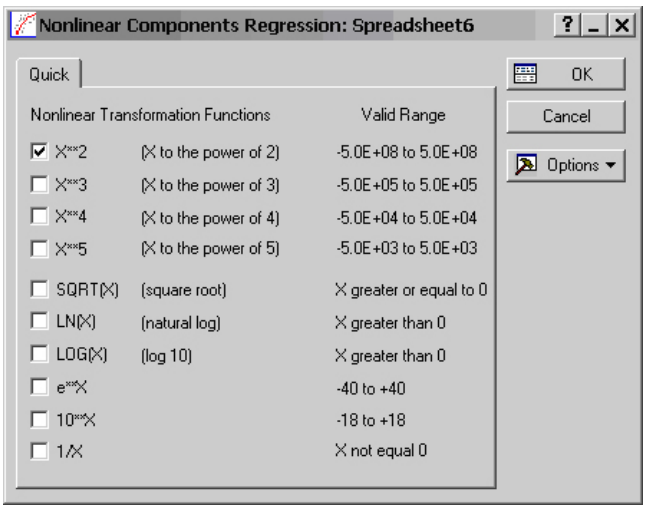


Рис. 1.3.3 – Обираємо квадратичну залежність

З’явиться вікно *Model Definition.* Перейдемо на закладку *Quick* і вкажемо залежну змінну – *VAR3*, а незалежними оберемо змінні *VAR1* та *VAR1**2* (див. рис. 1.3.4).

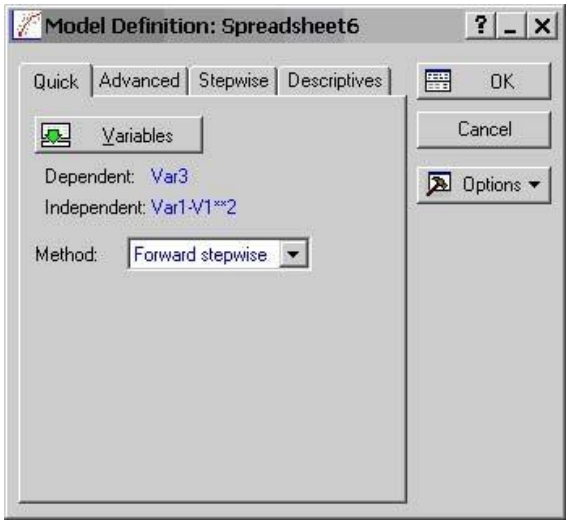


Рис. 1.3.4 – Налаштовуємо змінні

Будуємо регресійну модель покроково з відкиданнями незначимих змінних. У закладці *Stepwise* вказуємо значення 1.0 для *F to enter*, а у полі *F to remove* вказуємо 0.01. Як і очікували, у регресійній моделі одержали єдину значиму змінну *VAR1**2* (див. рис. 1.3.5).

Натиснувши *Summary: Regression results* отримуємо таблицю результатів

регресійного аналізу (див. рис. 1.3.6). Оцінки коефіцієнтів регресії близькі до значень у заданій теоретичній моделі.

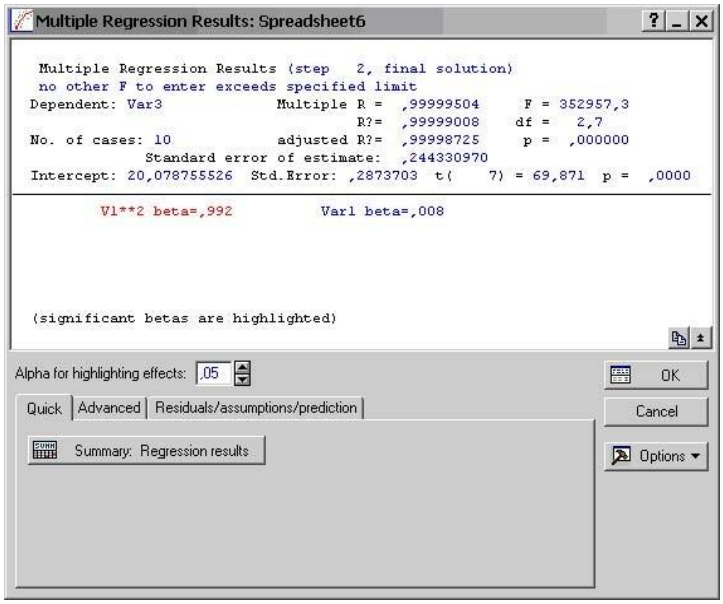


Рис. 1.3.5 – Будуємо регресійну модель

Regression Summary for Dependent Variable: Var3 (Spreadsh
R= ,99999504 R?= ,99999008 Adjusted R?= ,99998725
F(2,7)=3530E2 p<,00000 Std.Error of estimate: ,24433

N=10	Beta	Std.Err. of Beta	B	Std.Err. of B	t(7)	p-level
Intercept			20,07876	0,287370	69,8707	0,000000
V1**2	0,991750	0,005310	1,98585	0,010633	186,7608	0,000000
Var1	0,008458	0,005310	0,19117	0,120018	1,5928	0,155229

Рис. 1.3.6 – Таблиця результатів регресійного аналізу

Іноді з графічного зображення даних або попереднього досвіду роботи з ними можна зробити припущення про те, що для того щоб отримати адекватну регресійну модель, потрібно зробити перетворення залежної змінної вигляду:

$$\begin{cases} x^{\lambda}, \ \lambda \neq 0 \\ \ln(x), \ \lambda \neq 0. \end{cases}$$

Використаємо для цього допоміжну програму. Вона обчислить оцінку максимальної вірогідності параметра λ.

Для цього виконаємо послідовність команд *Help -> Open Examples -> Macros -> Analysis Examples* і викличемо програму *BoxCox*. У вікні, що з'явилось (див. рис. 1.3.7), бачимо текст програми на мові *STATISTICA Visual*

Basic.

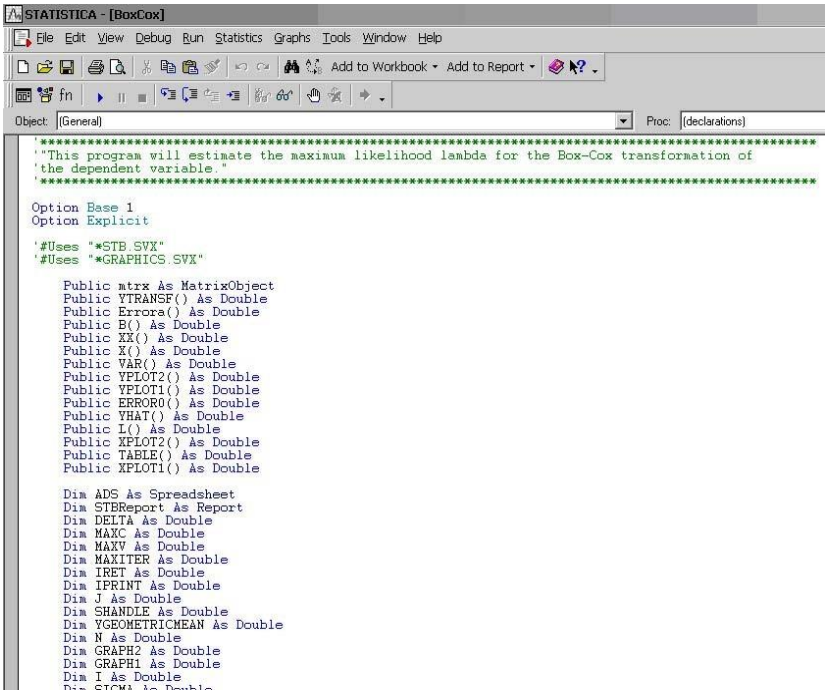


Рис. 1.3.7 – Обчислення оцінки максимальної вірогідності параметра λ

Натиснемо на стрілку в панелі макросів, щоб почати виконання програми. Вказуємо наші змінні і у вікні, яке потім з'являється (див. рис. 1.3.8), задаємо межі для параметра λ Натискаємо *OK*. Вибравши один із запропонованих методів (див. рис. 1.3.9) *Manual/visual search (graph of lambda vs. SSE)* або *Iterative optimization(golden search)* отримаємо оцінку для λ, яку потім можна використати при побудові регресійної моделі.

Рис. 1.3.8 – Межі для параметра λ

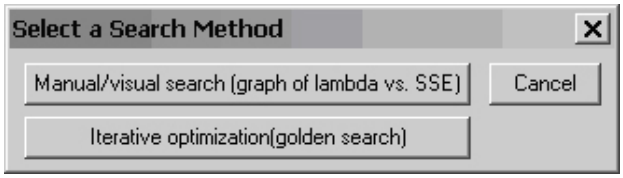


Рис. 9 – Вибір методів

За допомогою програми *Boxtidwell.stb*, яка знаходиться в тій же директорії, що і *BoxCox*, можна обчислити оцінки максимальної вірогідності для параметрів перетворення незалежних змінних.

Розглянемо деякі інші нелінійні регресійні моделі.

Відкриємо файл *program.sta* (див. рис. 1.3.10). У першій колонці файлу вказано витрачений працівниками час на вивчення певного програмного продукту, а у другій колонці вказано результати тестового завдання, на виконання якого було відведено фіксований час. Потрібно побудувати залежність успішності виконання від часу вивчення. Шукатимемо цю залежність у вигляді логістичної регресійної моделі:

$$y_t = \frac{\exp(b_0 + b_1 t)}{1 + \exp(b_0 + b_1 t)},$$

де y_t – успішність, t – час, b_0, b_1 - невідомі параметри.

Завантажимо модуль з нелінійного оцінювання: *Statistics -> Advanced/Linear Nonlinear Models -> Nonlinear Estimation* (див. рис. 1.3.11). Обираємо *Quick Logit regression* (див. рис. 1.3.12) і натискаємо *OK*.

	Success in a programin	
	1 EXPERNCE	2 SUCCESS
Frank	14	FAILURE
Henry	29	FAILURE
Tom	6	FAILURE
Beth	25	SUCCESS
Susan	18	SUCCESS
Harry	4	FAILURE
Paul	18	FAILURE
Pete	12	FAILURE
Diana	22	SUCCESS
Louise	6	FAILURE
Fred	30	SUCCESS
Hank	11	FAILURE
Steven	30	SUCCESS
Tod	5	FAILURE
Take	20	SUCCESS
Sam	13	FAILURE
Gail	9	FAILURE
Thomas	32	SUCCESS
Theodore	24	FAILURE
Charles	13	SUCCESS
Elizabeth	19	FAILURE
Lori	4	FAILURE
Ann	28	SUCCESS
Valerie	22	SUCCESS
Anke	8	SUCCESS

Рис. 1.3.10 – Файл program.sta

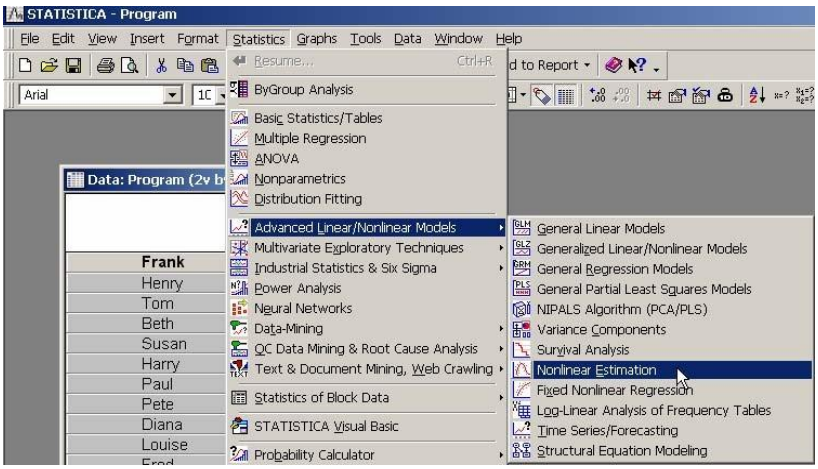


Рис. 1.3.11 – Завантаження модуля з нелінійного оцінювання

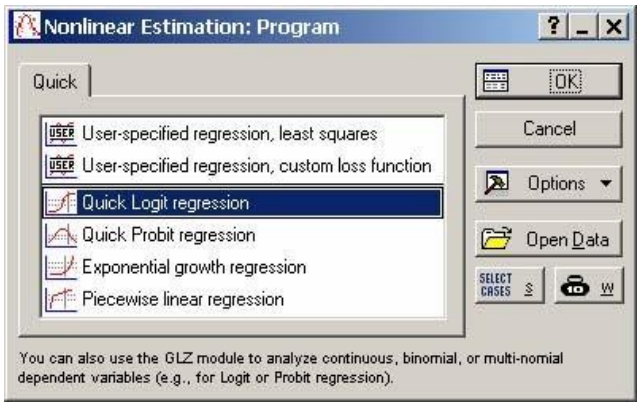


Рис. 1.3.12 – Обрання Quick Logit regression

Вкажемо як залежну змінну *Success*, а як незалежну – *Experience* (див. рис. 1.3.13). Натискаємо двічі *OK*. Отримані результати відобразяться у вікні, яке зображено на рисунку 1.3.14.

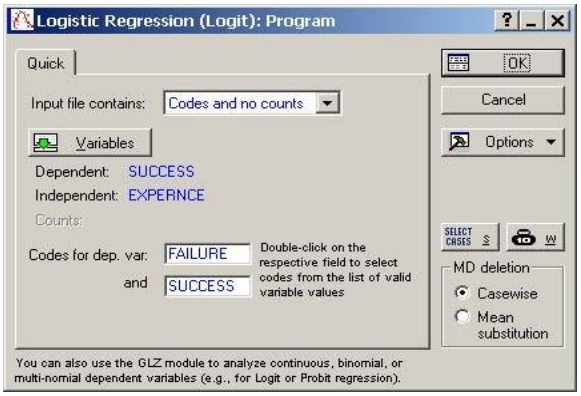


Рис. 1.3.13 – Встановлюємо залежні та незадежні змінні

Натиснувши *Fitted 2D function & observed values* отримаємо графік логістичної регресійної функції (див. рис. 1.3.15), яка найкраще описує наші дані. Формула, яка визначає функцію, написана над графіком. На цьому ж графіку зображено наші дані. Їхні координати по вертикальній осі є або 1, якщо тестове завдання виконано успішно, або 0, якщо ні. Використовуючи отриману логістичну модель можна передбачити наскільки успішно програміст впорається із завданням залежно від його досвіду роботи з програмним продуктом.

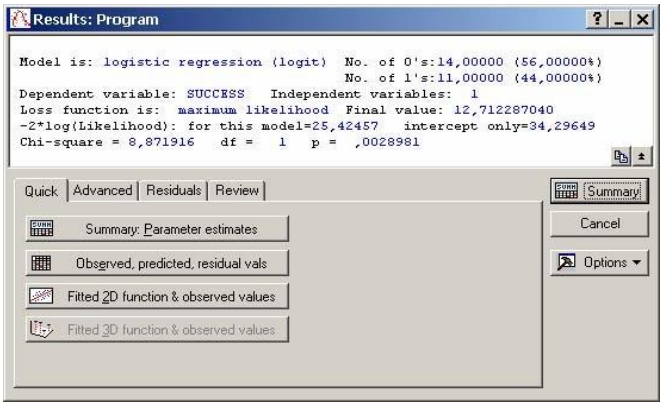


Рис. 1.3.14 – Отримані результати

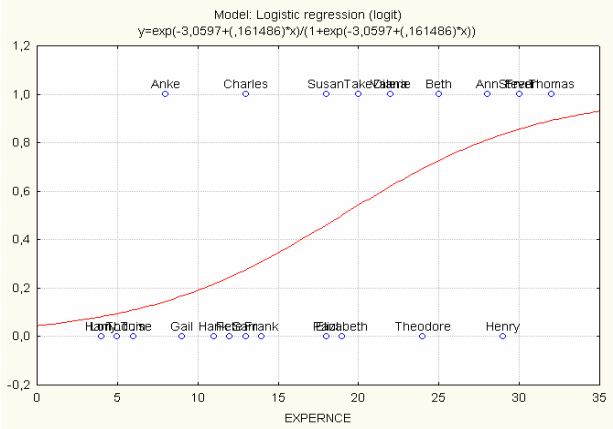


Рис. 1.3.15 – Графік логістичної регресії

Натискаючи різні кнопки у закладці *Residuals* (див. рис. 1.3.16), можемо отримати гістограму залишків, значення самих залишків, значення оцінок, які «пророкують» згідно із затраченим часом, вивчити відповідність розподілу залишків до гауссового розподілу. Наприклад і Q-Q графік і гістограма (див. рис.

1.3.17 і 1.3.18) свідчать про нормальний розподіл залишків.

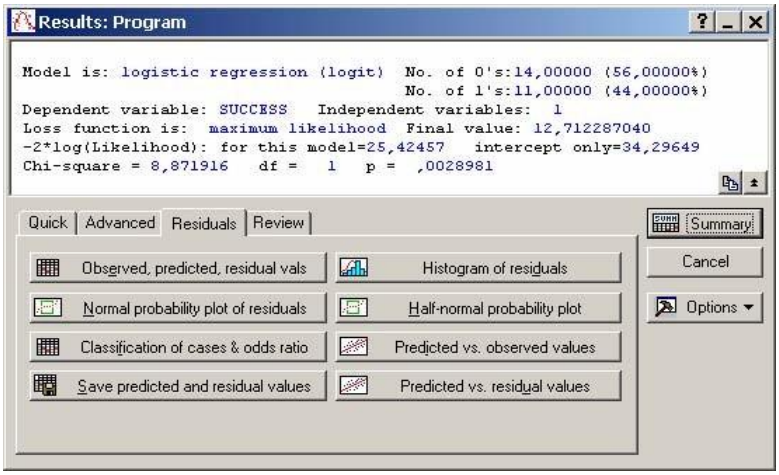


Рис. 1.3.16 – Закладка Residuals

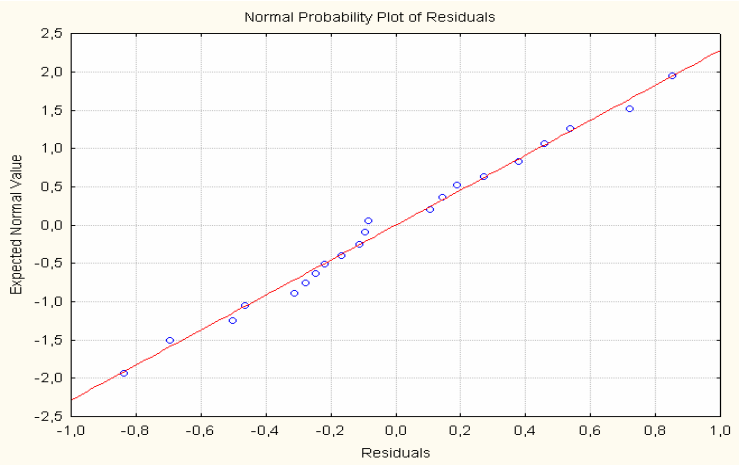


Рис. 1.3.17. – Q-Q графік

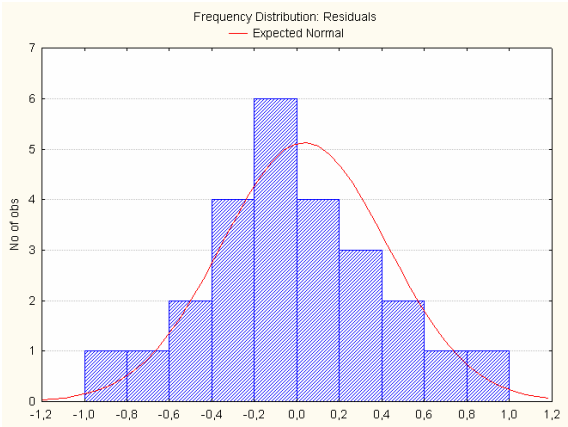


Рис. 1.3.18. – Q-Q гістограма

Відкриємо файл *Learning.sta* (див. рис. 1.3.19). Цей файл містить

інформацію, про два підприємства A і B, які переходять на випуск нової продукції. Одна змінна – час, інша ефективність – відношення прибутку на один виріб нової моделі до старої. З часом на підприємствах все краще освоюють технологію пов'язану з випуском нової продукції, тому ефективність зростає. Ми хочемо порівняти два підприємства між собою і визначити яке підприємство має більший потенціал для впровадження нових технологій. Для цього ми побудуємо регресійні моделі для даних кожного підприємства і порівняємо ці моделі.

Виконаємо *Statistics -> Advanced/Linear Nonlinear Models -> Nonlinear Estimation -> User specified regression, custom loss function* (див. рис. 1.3.12).

Як *Function to be estimated & loss function* задамо функцію

*v3=B +B *V1+B *exp(B *V2*),

0 1 3 2

а як міру відхилення *(OBS-PRED)**2* (див. рис. 1.3.20). Натискаємо *OK*.

Оскільки текстовому значенню першої змінної *Plant_A* відповідає 1, а текстовому значенню *Plant_B* відповідає 0, то додатний знак коефіцієнта *B* буде свідчити про те, що перше підприємство має більший потенціал для впровадження нових технологій, а якщо $B < 0$, то навпаки - друге.

	Relative efficiency of production		
	1 LOCATION	2 WEEKS	3 EFFICNCY
1	PLANT_A	1	0,483
2	PLANT_A	2	0,539
3	PLANT_A	3	0,618
4	PLANT_A	5	0,707
5	PLANT_A	7	0,762
6	PLANT_A	10	0,815
7	PLANT_A	15	0,881
8	PLANT_A	20	0,919
9	PLANT_A	30	0,964
10	PLANT_A	40	0,959
11	PLANT_A	50	0,968
12	PLANT_A	60	0,971
13	PLANT_A	70	0,960
14	PLANT_A	80	0,967
15	PLANT_A	90	0,975
16	PLANT_B	1	0,517
17	PLANT_B	2	0,598
18	PLANT_B	3	0,635
19	PLANT_B	5	0,750
20	PLANT_B	7	0,811
21	PLANT_B	10	0,848
22	PLANT_B	15	0,943
23	PLANT_B	20	0,971
24	PLANT_B	30	1,012

Рис. 1.3.19 – Файл Learning.sta

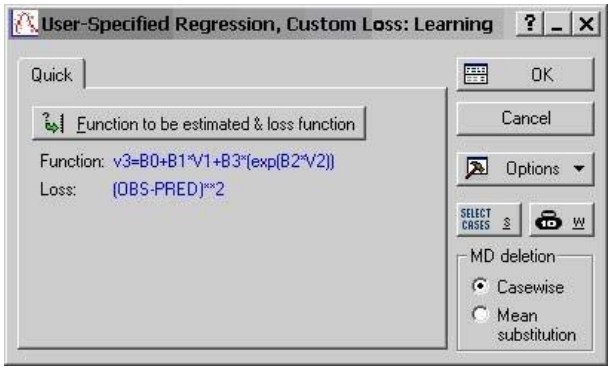


Рис. 1.3.20 – Задання функції та міри відхилення

У закладці *Advanced* як *Estimation method* виберемо *Rosenbrock and quasi-Newton*. Як *Start values* задамо всі значення 0 (див. рис. 1.3.21). Зауважимо, що успіх побудови моделі залежить від вдалого вибору початкових значень та кроку зміни параметрів. Натискаємо *OK*.

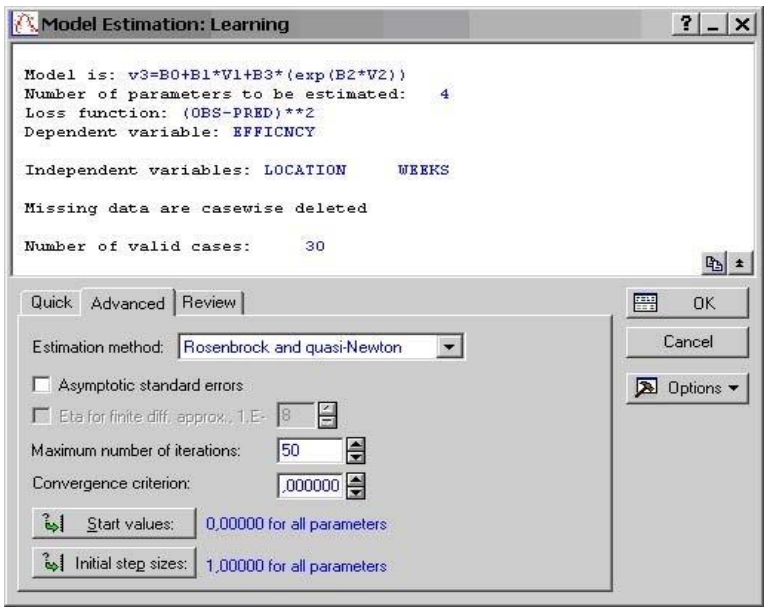


Рис. 1.3.21 – Задання початкових значень

У вікні *Results*, що з’явилось (див. рис. 1.3.22), бачимо результати аналізу. Значення *Proportion of variance accounted for* показує наскільки пояснено розкид даних.

Побачити оцінки для параметри моделі можемо, натиснувши кнопку

Summary (див. рис. 1.3.23). Оскільки $B < 0$, то друге підприємство має більший потенціал для впровадження нових технологій. Це очевидно, якщо подивитись на малюнок 19 з даними, які ми використовували для нашого дослідження.

Аналіз залишків, як і раніше, можна здійснити використовуючи меню з закладки *Residuals* (див. рис. 1.3.22). У результаті приходимо до висновку, що модель добре описує наявні дані.

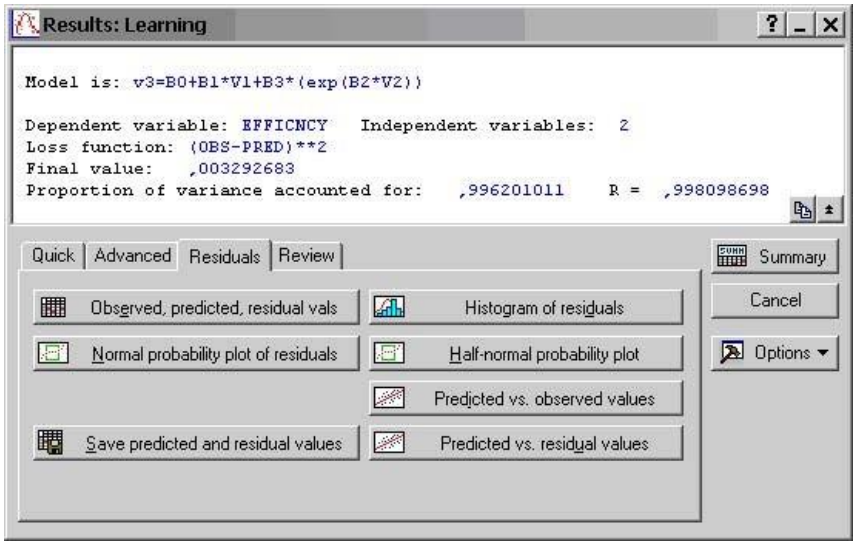


Рис. 1.3.22 – Аналіз залишків

Model: v3=B0+B1*V1+B3*(exp(B2*V2)) (Learning)
Dep. var: EFFICNCY Loss: (OBS-PRED)**2
Final loss: ,003292683 R= ,99810 Variance explained: 99,620%

N=30	B0	B1	B3	B2
Estimate	1,015598	-0,047267	-0,552442	-0,134795

Рис. 1.3.23 – Отримані результати

Висновки:

У даній лабораторній роботі:

– було описано види регресії;

– видобування знань за допомогою регресійного аналізу;

1.4 Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Текст розробленого програмного забезпечення з коментарями, а також текст програми для тестування розроблених генетичних методів.
4. Графіки та аналітичні вирази обраних тестових функцій.
5. Результати роботи програмного забезпечення.
6. Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх аналіз.

1.5 Контрольні питання

1. Як створити новий документ з трьома змінними? Як заповнити змінну послідовно значеннями 1, 2, 3, 4, 5 і т.д. Як заповнити змінну випадковими величинами від 0 до 1? (Опишіть послідовність дій)
2. Як виключити з розрахунку і побудови графіків викиди?
3. Як можна забезпечити, щоб значення викиду не виводилось у наступних графіках та не враховувалось при обчисленні регресійної формули?
4. Як знайти формулу багатофакторної лінійної регресії? У чому полягає відмінність між однофакторною і багтофакторною регресією?
5. На рис. 4. представлені результати аналізу. Яким чином з рисунка можна визначити які змінні несуттєво впливають на регресійну модель?
6. На основі яких результатів можна вважати, що багатофакторна регресійна модель достатньо добре описує дані?
7. Скільки незалежних змінних розглядається в прикладі *Job_prof.sta ?*
8. Наведіть приклад мультиколінеарності змінних.
9. Як завантажити модуль з нелінійного оцінювання в Statistica?
10. Як отримати графік логістичної регресійної функції?

2 ЛАБОРАТОРНА РАБОТА №2. МЕТОДИ КЛАСТЕРИЗАЦІЇ. КЛАСТЕРНИЙ АНАЛІЗ. ДЕРЕВА РІШЕНЬ

2.1 Мета роботи

Ознайомилися з програмним пакетом **Waikato Environment for Knowledge Analysis (WEKA)**, що дозволяє аналізувати значні обсяги інформації та визначати існуючі в них тренди і залежності. Ознайомлення з методами класифікації та кластеризації.

2.2 Методичні рекомендації до самостійної роботи студентів

В попередній лабораторній роботі ми отримали загальне уявлення про основні принципи інтелектуального аналізу даних. Крім того, ми ознайомилися з одним з методів інтелектуального аналізу даних, а саме **регресійним аналізом,** за допомогою якого можна спрогнозувати числове значення залежної величини на підставі набору показників незалежних величин.

У цій лабораторній роботі ми ознайомимося з двома іншими методами інтелектуального аналізу даних. За своєю структурою ці методи значно складніші, ніж регресійний аналіз, проте їх використання дозволяє досягти значних результатів. Якщо при використанні регресійного аналізу ви просто отримуєте якесь числове значення, залежне від вхідних параметрів, то застосування двох інших методів дозволить вам по-новому інтерпретувати отримані дані. Завдання інтелектуального аналізу - визначити і побудувати модель, відповідну нашим даним. Наприклад, ми можемо зібрати велику колекцію даних про клієнтів, але якщо у нас немає правильної моделі для інтерпретації цих даних, то вся зібрана нами інформація нічого не варта.

Поглянемо на це питання по-іншому: якби всі використовували тільки регресійний аналіз, то як той чи інший інтернет-магазин зміг би сказати нам: «Покупці, які купили X, також купили і Y»? Числової функції, яка вміла б

визначати подібну інформацію, просто не існує. Тому ознайомимося з двома новими методами аналізу і розберемося, яку інформацію ми можемо отримати з їх допомогою.

Тум ми будемо досить часто зустрічати посилання на метод найближчих сусідів без пояснень суті цього методу. Ми більш детально розглянемо його в майбутньому, а поки будемо використовувати його для порівняння при описі методів, розглянутих у цій лабораторній роботі. Порівняльний аналіз цих методів допоможе нам краще зрозуміти їхні можливості.

Класифікація, кластеризація і метод найближчих сусідів

Перш ніж ми перейдемо до обговорення конкретних деталей кожного з розглянутих методів та їх реалізації в WEKA, давайте розберемося, для чого використовується цей метод, якими даними він оперує і яку інформацію можна отримати з його допомогою .

Для порівняльного аналізу методів ми скористаємося реальними прикладами дилерського центру BMW і його способи підвищення продажів. Дилерський центр зберігає всі дані про своїх клієнтів, включаючи тих, хто купив BMW, цікавився чи просто зайшов в демонстраційній зал BMW. Мета дилерського центру - підвищити рівень продажів, спираючись на результати інтелектуального аналізу отриманих даних.

Класифікація

Запитання: «Яка ймовірність того, що хтось купить останню модель BMW M5?» Створення класифікаційної моделі (дерева рішень) дозволяє оцінити ймовірність того, що якась людина купить автомобіль моделі M5. В якості вузлів дерева можуть використовуватися такі показники, як вік, рівень доходу, кількість вже наявних машин, сімейний стан, діти, наявність власного будинку або оренда будинку. Параметри конкретної людини будуть використовуватися для проходження по дереву з метою визначення ймовірності покупки M5.

Кластеризація

Запитання: «Який вік покупців, що віддають перевагу сріблястий BMW M5?». Необхідні дані можуть бути отримані при аналізі віку покупців і кольору

обраних ними машин. Далі, на підставі отриманих даних, можна зробити висновок, що певна вікова група (наприклад, люди у віці від 22 до 30 років) віддає перевагу певному кольру BMW M5 (75% купують автомобілі синього кольору). Точно так, можна визначити, що інша вікова група (наприклад, люди у віці від 55 до 62 років) віддає перевагу сріблястому BMW (65% купують сріблясті автомобілі, а 20% купують сірі). Аналіз даних дозволить створити кластери за віковими групами і кольорами і визначити залежності між даними.

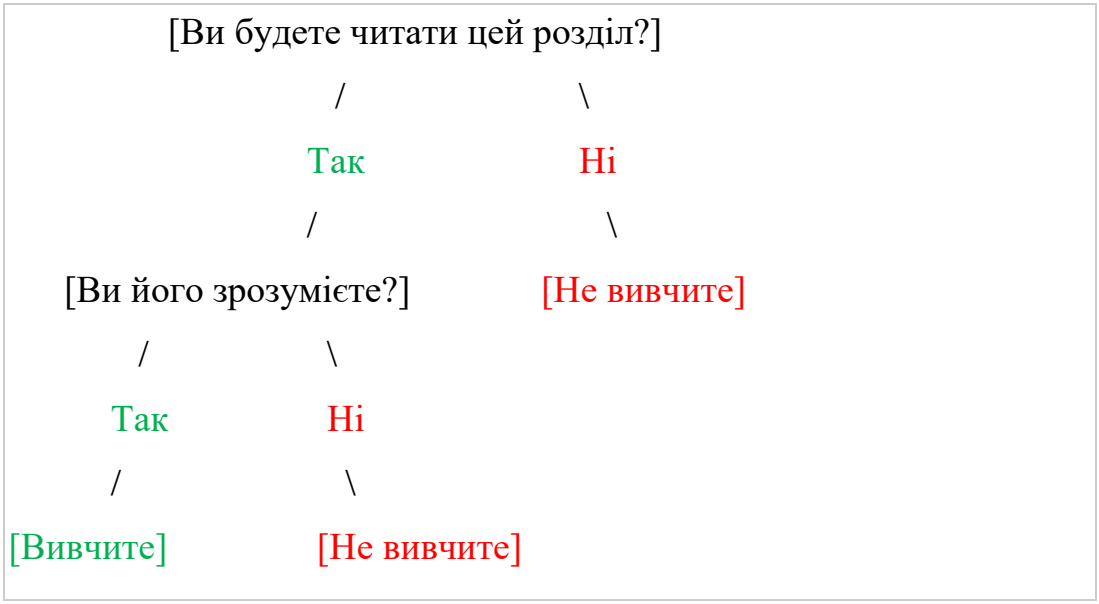
Метод найближчих сусідів

Питання : «Якщо людина купує автомобіль BMW M5, які ще товари вона може придбати?» Як свідчать статистичні дані, якщо людина купує BMW M5, то досить часто вона бере відповідну за кольором барсетку (подібні дослідження називаються «аналіз ринкового кошика»). Використовуючи подібні дані, дилерський центр може розмістити рекламу відповідних товарів або подати на друк оголошення про знижки на барсетки і сумки відповідного кольору або їх безкоштовну пропозицію всім покупцям M5 в якості одного із способів підвищення кількості продажів.

Класифікація

Метод класифікації (також відомий як метод класифікаційних дерев або дерев прийняття рішень) - це алгоритм аналізу даних, який визначає покроковий спосіб прийняття рішення в залежності від значень конкретних параметрів. Дерево цього методу має наступний вигляд: кожен вузол являє собою точку прийняття рішення на підставі вхідних параметрів. Залежно від конкретного значення параметра ви переходите до наступного вузла, від нього - до наступного вузла, і так далі, поки не дійдете до листка, який і дає вам остаточне рішення. Звучить досить заплутано, але насправді метод досить прямолінійний. Давайте звернемося до конкретного прикладу.

Просте класифікаційне дерево



Це просте класифікаційне дерево визначає відповідь на питання «Чи вивчите ви принцип побудови класифікаційного дерева?» У кожному вузлі ви відповідаєте на відповідне питання і переходьте за відповідній гілці до наступного вузла, до тих пір, поки не дійдете до листа з остаточною відповіддю «так» чи «ні». Ця модель застосовна до будь-яких сутностей, і ви зможете відповісти, чи в змозі ці сутності вивчити класифікаційні дерева, за допомогою двох простих питань. У цьому і полягає основна перевага класифікаційних дерев - вони не вимагають надмірної кількості інформації для створення досить точного і інформативного дерева рішень.

Модель класифікаційного дерева використовує підхід, аналогічний тому, який ми застосовували при створенні моделі регресійного аналізу, а саме створення моделі на основі навчальної послідовності (training set). Цей підхід **використовує відомі дані** для аналізу впливу **вхідних значень** на **результат** і будує модель на основі отриманих залежностей. **Таким чином, коли у нас є новий набір даних з невідомим результатом, ми підставляємо наші дані в модель і отримуємо очікуваний висновок.**

Поки що новий метод працює так само, як регресійна модель. Проте, існують і відмінності. Метод класифікаційного аналізу використовує покращений підхід, суть якого полягає в тому, що набір відомих даних ділиться

на дві частини. **Перша частина** (60-80% даних) використовується в якості навчального набору для створення моделі. Після цього дані, що залишилися проганяються через цю модель, і змодельовані результати порівнюються з реальними. Такий підхід дозволяє оцінити точність аналітичної моделі.

Для чого потрібен додатковий крок перевірки моделі? Він дозволяє уникнути *зайвої підгонки* моделі під зібрані дані. Якщо ми будемо використовувати *занадто багато* даних для розробки моделі, то отримана модель буде ідеально відповідати зібраними даними і тільки їм. Наше завдання - створити модель для прогнозування невідомих нам даних, а не для прогнозування даних, які і так вже відомі. Для перевірки цього і використовується тестовий набір. З його допомогою ми визначаємо, що точність моделі на тестовому наборі не зменшується. Таким чином, ми гарантуємо, що наша модель зможе з досить високою ймовірністю визначити невідомі значення. З допомогою **WEKA** ми розглянемо, як це працює на конкретному прикладі.

Питання надмірної кількості даних приводить нас до обговорення ще одного важливого принципу побудови класифікаційних дерев - принципу відсікання гілок. *Відсікання гілок*, як випливає з назви, має на увазі видалення гілок з класифікаційного дерева. Але навіщо видаляти інформацію з дерева рішень? Знову ж, для того, щоб уникнути зайвої підгонки дерева під відомі дані. У міру зростання даних зростає і кількість атрибутів, так що наше дерево може стати надмірно складним і розгалуженим. Теоретично, можна створити дерево з кількістю листків leaves = (rows * attributes) , проте наскільки корисне таке дерево? Воно навряд чи зможе допомогти нам визначити невідомий результат, так як воно всього лише визначає вже відомі дані. Необхідно досягти певного балансу між точністю аналізу і простотою моделі. Нам потрібна аналітична модель з мінімальною кількістю вузлів і листків, яка, тим не менш, відрізнялася б високою точністю. Тут, як ми бачимо, необхідний певний компроміс.

І останнє питання, на якому ми хотіли б зупинитися, перш ніж приступити до створення конкретних моделей в WEKA, це **проблема помилкового розпізнавання** (false positive і false negative). Основний сенс проблеми помилкового розпізнавання полягає в тому, що модель розглядає який-небудь атрибут який має вплив на кінцевий результат, в той час як насправді цей атрибут ніякого впливу на результат не надає. І навпаки, істотний атрибут трактується моделлю як несуттєвий.

Хибне розпізнавання свідчить про помилковість моделі і невірну класифікацію даних. Безумовно, певна похибка в класифікації припустима, і завдання розробника моделі - визначити допустимий відсоток помилок. Наприклад, при розробці моделі оцінки кардіомоніторів для лікарень від вас будуть потрібні виключно низькі показники помилковості результатів. Навпаки, якщо ви розробляєте навчальну модель для ілюстрації статті про інтелектуальний аналізі даних, ви можете дозволити собі достатньо високий рівень похибки моделі. Розвиваючи цю тему, корисно визначити прийнятне процентне відношення хибно-негативного розпізнавання до хибно-позитивного. Найбільш очевидний приклад - модель визначення спаму. Хибно-позитивне розпізнавання (потрібний лист позначається як спам), швидше за все, буде мати більше негативних наслідків, ніж хибно-негативні (лист-спам потрапить в розряд потрібних листів). У цьому випадку може вважатися прийнятним співвідношення хибно-негативних розпізнавань до хибно-позитивних як 100:1.

2.3 Порядок виконання роботи

Тепер візьмемо реальні дані і пропустимо їх через аналітичний механізм WEKA.

Набір даних, який ми будемо використовувати для прикладу класифікаційного аналізу, містить інформацію, зібрану дилерським центром BMW. Центр починає рекламну кампанію, пропонуючи розширену дворічну

гарантію своїм постійним клієнтам. Подібні кампанії вже проводилися, так що дилерський центр має 4500 показники щодо попередніх продажів з розширеною гарантією. Цей набір даних має такі атрибути:

- Розподіл за доходами
  [0=$0-$30k,
  1=$31k-$40k,
  2=$41k-$60k,
  3=$61k-$75k,
  4=$76k-$100k,
  5=$101k-$150k,
  6=$151k-$500k,
  7=$501k+]
- Рік / місяць покупки першого автомобіля BMW
- Рік / місяць покупки останнього автомобіля BMW
- Чи скористався клієнт розширеною гарантією

Файл даних у форматі Attribute-Relation File Format (ARFF) буде виглядати наступним чином:

```
@attribute IncomeBracket {0,1,2,3,4,5,6,7}
@attribute FirstPurchase numeric
@attribute LastPurchase numeric
@attribute responded {1,0}

@data

4,200210,200601,0
5,200301,200601,1
...
```

Класифікація в WEKA

Завантажте файл **bmw-training.arff** в програмний пакет **WEKA**, використовуючи ті ж кроки, які ми виконали для завантаження даних у випадку регресійного аналізу.

Зауваження: *в пропонованому файлі містяться 3000 з наявних 4500 записів. Ми розділили набір даних так, щоб частина їх використовувалася для створення моделі, а частина - для перевірки її точності, щоб переконатися, що модель не є підігнаною під конкретний набір даних.*

Після завантаження даних вікно WEKA має виглядати так, як показано на рис. 2.1.

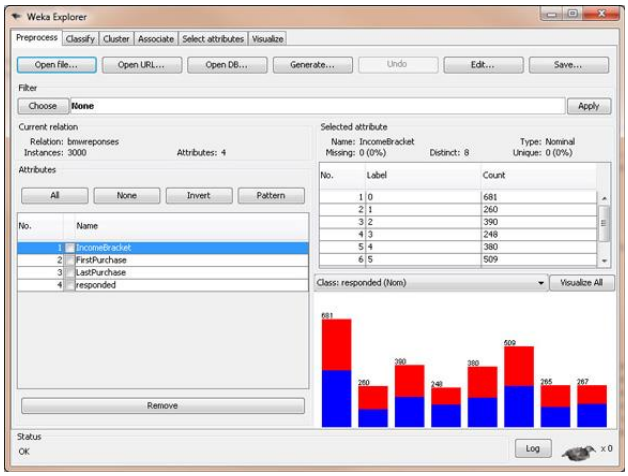


Рис. 2.1 – Дані дилерського центру BMW для класифікації в WEKA

Аналогічно тому, як ми вибирали тип моделі для регресійного аналізу в попередній лабораторній роботі, тепер нам слід вибрати модель для класифікації: відкрийте закладку **Classify** , виберіть опцію **trees**, а потім опцію **J48**.

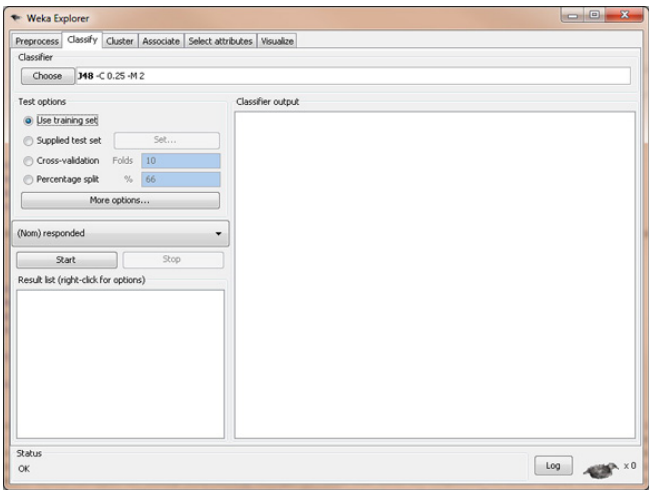


Рис. 2.2 – Алгоритм класифікації даних BMW

Тепер ми готові приступити до створення конкретної моделі аналізу засобами пакета WEKA. Переконайтеся, що обрана опція **Use training set** , щоб пакет WEKA при створенні моделі використовував саме ті дані, які ми тільки що завантажили у вигляді файлу. Натисніть кнопку **Start** і надайте WEKA можливість попрацювати з нашими даними. Результуюча модель повинна виглядати так, як показано нижче:

Результат роботи класифікаційної моделі WEKA

```
Number of Leaves  :     28

Size of the tree :      43


Time taken to build model: 0.18 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances        1774               59.1333 %
Incorrectly Classified Instances      1226               40.8667 %
Kappa statistic                          0.1807
Mean absolute error                      0.4773
Root mean squared error                  0.4885
Relative absolute error                 95.4768 %
Root relative squared error             97.7122 %
Total Number of Instances             3000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.662    0.481      0.587     0.662     0.622      0.616    1
                 0.519    0.338      0.597     0.519     0.555      0.616    0
Weighted Avg.    0.591    0.411      0.592     0.591     0.589      0.616

=== Confusion Matrix ===

   a    b   <-- classified as
 1009  516 |    a = 1
 710   765 |    b = 0
```

Розберемося що означають всі ці числа. Як нам зрозуміти, наскільки хороша отримана модель? І де, власне, це так зване «дерево», яке мало вийти в результаті? Цілком закономірні питання. Давайте відповімо на кожне з них по черзі:

Що означають всі ці числа? Найбільш суттєві дані - це показники класифікації "**Correctly Classified Instances**" (59.1%) і "**Incorrectly Classified Instances**" (40.9%). Крім того, слід звернути увагу на число в першому рядку стовпця **ROC Area** (0.616). Трохи пізніше ми докладніше обговоримо ці значення, поки ж просто запам'ятайте їх. Нарешті, таблиця **Confusion Matrix** показує кількість **хибно-позитивних** (516) і **хибно-негативних** (710) розпізнавань.

Як зрозуміти, наскільки хороша отримана модель? Оскільки показник точності нашої моделі - 59,1%, то в первісному розгляді її не можна назвати досить хорошою.

Де це так зване дерево? Ви зможете побачити дерево, якщо клацнете правою кнопкою миші в панелі результуючої моделі. У контекстному меню виберіть опцію **Visualize tree**. На екрані відобразиться візуальне уявлення класифікаційного дерева нашої моделі (рис. 2.3), проте в даному випадку картинка мало чим нам допоможе. Ще один спосіб побачити дерево моделі - прокрутити вгору висновок у вікні **Classifier Output**, там ви знайдете текстовий опис дерева з вузлами і листками.

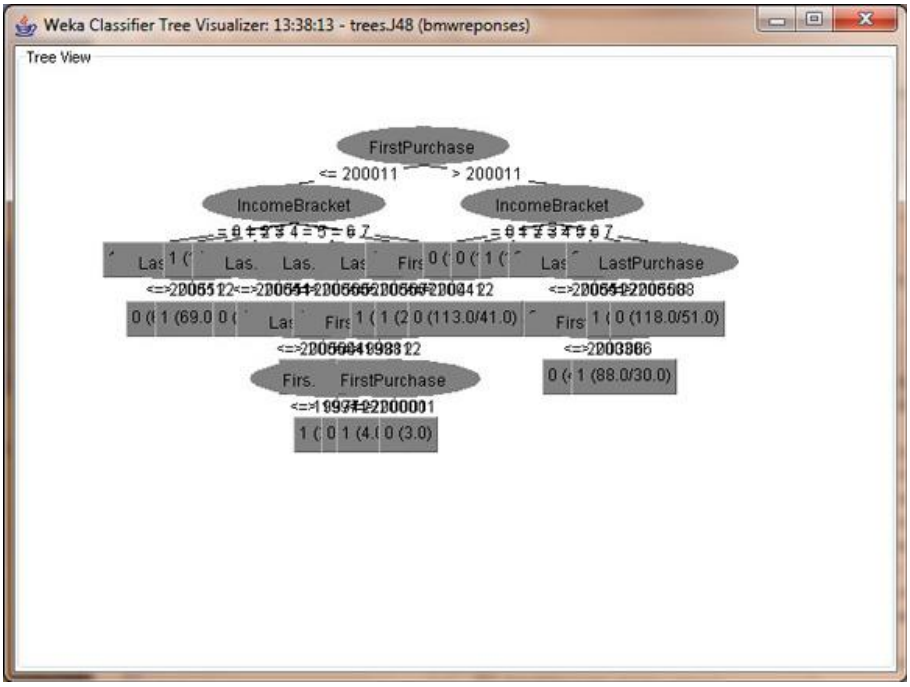


Рис.2.3 – Візуальне подання дерева класифікації

Залишився останній етап перевірки класифікаційного дерева: нам треба пропустити набір даних, що залишився через отриману модель і перевірити, наскільки результати класифікації будуть відрізнятися від реальних даних. Для цього в секції **Test options** виберіть опцію **Supplied test set** і натисніть на кнопку **Set**. Вкажіть файл **bmw-test.arff**, що містить решту 1500 даних, які не були включені в навчальний набір. При натисканні на кнопку **Start** WEKA пропустить тестові дані через модель і покаже результат роботи моделі. Давайте натиснемо на **Start** і перевіримо, що у нас вийшло.

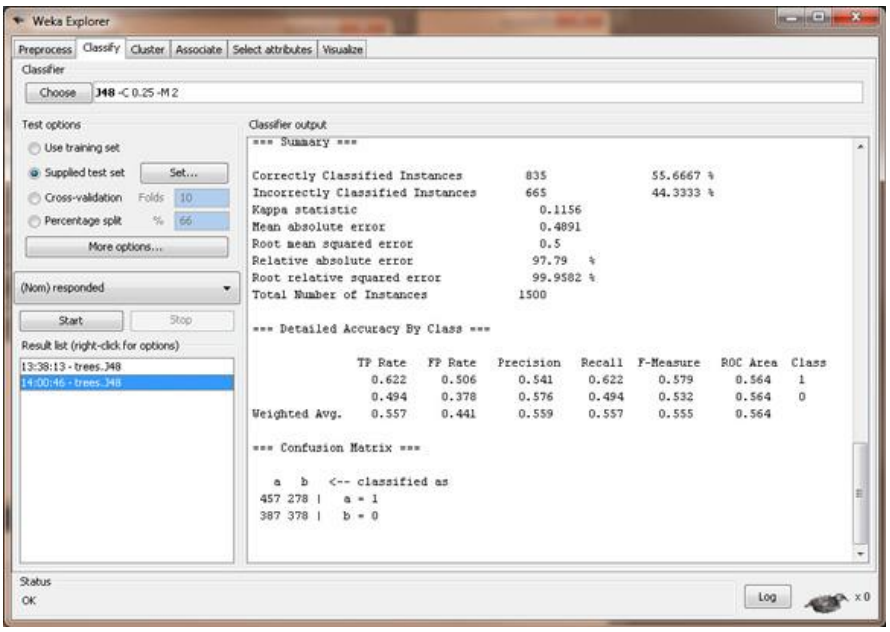


Рис 2.4 – Перевірка класифікаційного дерева

Порівнюючи показник **Correctly Classified Instances** для тестового набору (55,7%) з цим же показником для навчального набору (59,1%), ми бачимо, що точність моделі для двох різних наборів даних приблизно однакова. Це означає, що нові дані, які будуть використовуватися в цій моделі в майбутньому, не знизять точність її роботи.

Однак, оскільки власне точність моделі досить низька (всього лише 60% даних класифіковано вірно), ми маємо повне право зупинитися і сказати: «На жаль, ця модель взагалі нікуди не годиться. Вона працює з точністю трохи вище 50%, з таким же успіхом ми можемо просто намагатися вгадати значення випадковим чином». І цей вислів буде цілком справедливим. Це приводить нас до розуміння дуже важливого факту: існують випадки, коли використання алгоритмів інтелектуального аналізу даних призводить до створення невдалої аналітичної моделі. Наш приклад - наочнй тому доказ, і саме для цього ми його розглянули.

Ми свідомо проробили всі кроки, необхідні для створення класифікаційного дерева на підставі даних, здавалося б, ідеально відповідних для класифікаційної моделі. Однак, результат, отриманий WEKA, вказує на помилковість наших міркувань. Класифікаційна модель не підходить для аналізу

наявних у нас даних. Створена нами модель не дасть нам ніяких корисних відомостей, а її використання може призвести до прийняття неправильних рішень і втрати грошей.

Чи означає це, що наші дані взагалі не підлягають ніякому аналізу? Відповідь демонструє ще одну важливу особливість інтелектуального аналізу даних: використовуючи метод «найближчих сусідів», який ми детально розглянемо в майбутньому, ми створимо іншу модель на базі цього ж набору даних, з точністю роботи 88%. Отже, завжди необхідно пам'ятати, що **для того, щоб витягти корисну інформацію з великого набору даних, вам слід вибрати відповідну модель.**

Кластеризація

Кластеризація дозволяє розбити дані на групи, кожна з яких має певні ознаки. Метод кластерного аналізу використовується в тих випадках, коли необхідно виділити деякі правила, взаємозв'язки або тенденції у великих наборах даних. Залежно від потреб, ви можете виділити кілька різних груп даних. Одна з явних переваг кластеризації в порівнянні з класифікацією полягає в тому, що для розбиття множини на групи може використовуватися будь-який атрибут (метод класифікації використовує тільки певну підмножину атрибутів). В якості основного недоліку методу кластеризації слід зазначити той факт, що укладач моделі повинен заздалегідь вирішити, на скільки груп слід розбити дані. Для людини, яка не має жодного уявлення про конкретний набір даних, прийняти таке рішення досить важко. Нам варто створити три групи або п'ять груп? А може, нам потрібно визначити десять груп? Може знадобитися кілька повторювань проб і помилок, для того щоб визначити оптимальну кількість кластерів.

Тим не менше, для середньостатистичного користувача кластеризація може виявитися найбільш корисним методом інтелектуального аналізу даних. Цей метод дозволить вам швидко розбити ваші дані на окремі групи і зробити конкретні висновки і припущення щодо кожної групи. Математичні методи, що реалізують кластерний аналіз, досить складні і заплутані, так що в

разі кластеризації ми будемо цілком покладатися на обчислювальні можливості WEKA.

Короткий огляд математичних методів

Далі пропонується короткий загальний опис математичних методів і алгоритмів, використовуваних методом кластеризації.

1. Кожен атрибут має бути приведений до нормального вигляду. Для цього кожен показник ділиться на різницю між найбільшим і найменшим значенням, які приймає розглянутий атрибут на конкретному наборі даних. Наприклад, якщо розглянутий атрибут - вік, і його найбільше значення - 72, а найменше - 16, то значенням 32 буде відповідати нормалізована величина 0.5714.
2. Виходячи з бажаної кількості кластерів, випадковим чином вибирається така ж кількість рядків даних. Ці рядки будуть використовуватися в якості початкових центрів мас кластерів.
3. Для кожного рядка даних визначається відстань від цього рядка до центру мас кластера (випадковим чином вибраного рядка даних) за допомогою методу найменших квадратів.
4. Кожен рядок набору даних входить у той кластер, відстань до центру мас якого виявилося найменшим.
5. У кожному кластері визначається новий центр мас як набір середніх значень по стовпцях на безлічі елементів цього кластеру.
6. Визначається відстань від кожного елемента даних до нового центру мас. Якщо при цьому розподіл елементів по кластерах не змінюється, розбиття даних на кластери закінчено, і всі групи даних визначені. Якщо склад кластерів змінюється, слід повернутися до п.3 і повторювати цей процес до тих пір, поки розбиття на кластери не стане незмінним.

Для того щоб розбити набір даних з 10 рядків на три кластери за допомогою електронних таблиць, потрібно приблизно півгодини напруженої роботи. Можна уявити, скільки часу займе розбиття на 10 кластерів набору з 100000 записів. Але комп'ютер здатний виконати подібні розрахунки за кілька секунд.

Набір даних для WEKA

Для побудови моделі кластеризації ми знову скористаємося даними дилерського центру BMW. Співробітники центру зібрали дані про усіх відвідувачів демонстраційного залу, машинах, які їх зацікавили, і про те, наскільки часто відвідувачі демонстраційного залу в підсумку купували автомобіль, який їм сподобався. Тепер дилерському центру треба проаналізувати ці дані для того, щоб виділити різні групи відвідувачів і зрозуміти, чи не можна визначити будь-які тенденції в їх поведінці. У нашому прикладі використовується 100 записів, і кожен стовпець описує певний етап, який, як правило, проходить покупець у процесі вибору та придбання автомобіля. Відповідно, значення 1 у стовпці говорить про те, що відвідувач пройшов конкретний етап, а 0 - що відвідувач цей етап не пройшов. Файл з даними у форматі ARFF приведений нижче.

Дані для кластерного аналізу засобами WEKA

```
@Attribute Dealership numeric
@Attribute Showroom numeric
@Attribute ComputerSearch numeric
@Attribute M5 numeric
@Attribute 3Series numeric
@Attribute Z4 numeric
@Attribute Financing numeric
@Attribute Purchase numeric

@Data
1,0,0,0,0,0,0,0
1,1,1,0,0,0,1,0
...
```

Кластеризація в WEKA

Завантажте файл **bmw-browsers.arff** в WEKA, виконавши ті ж кроки, які ми виконали раніше для відкриття даних у закладці **Preprocess**. Витратьте кілька хвилин на візуальний аналіз даних, зверніть увагу на атрибути даних, розподіл

даних по стовпцях, і так далі. Після завантаження даних ваш екран WEKA повинен виглядати так, як показано на рис. 2.5.

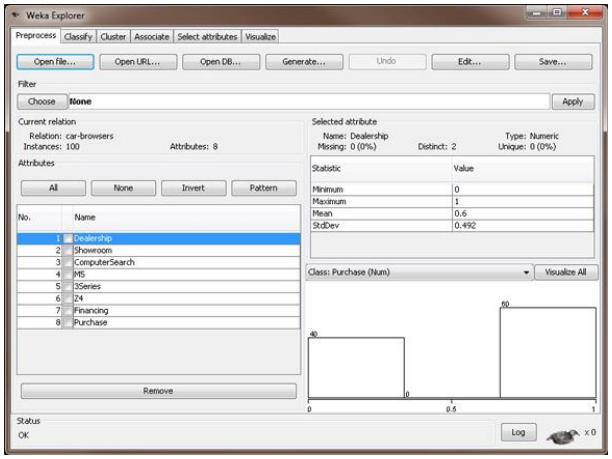


Рис. 2.5 – Дані BMW для кластеризації

Оскільки ми хочемо розбити наявні в нас дані на кластери, замість закладки **Classify** нам буде потрібно закладка **Cluster**. Натисніть на кнопку **Choose** і в пропонованому меню виберіть опцію **SimpleKMeans** (в рамках даної роботи ми будемо користуватися цим методом кластеризації). У результаті вікно **WEKA Explorer** буде виглядати так, як показано на рис. 2.6.

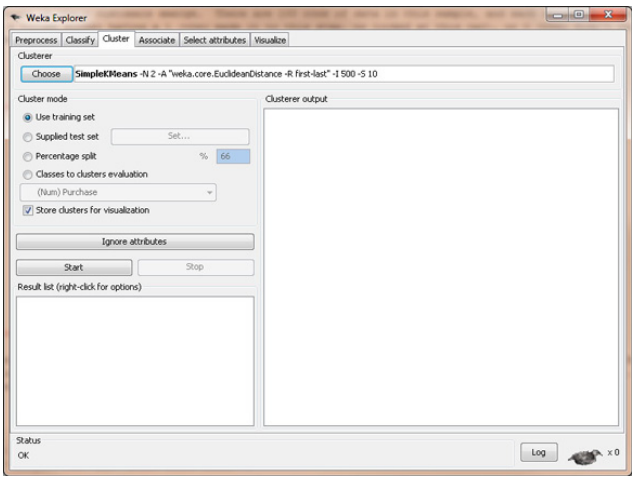


Рис. 2.6 – Алгоритм кластеризації даних BMW

Тепер нам потрібно вибрати необхідні параметри для нашого алгоритму кластеризації. Клацніть на опції **SimpleKMeans** (дизайн користувальницького інтерфейсу залишає бажати кращого, але працювати з ним можна). Єдиний атрибут алгоритму, який нас цікавить - це поле **numClusters**, яке вказує на

кількість кластерів для розбиття (нагадуємо, що це значення вам потрібно вибрати ще до створення моделі). Змінимо значення за замовчуванням (**2**) на **5**. Постарайтеся запам'ятати послідовність кроків, щоб ви змогли згодом змінити кількість кластерів. Тепер ваше вікно **WEKA Explorer** має виглядати так, як показано на рис. 2.7. Натисніть на кнопку **OK**, щоб зберегти вибрані параметри.

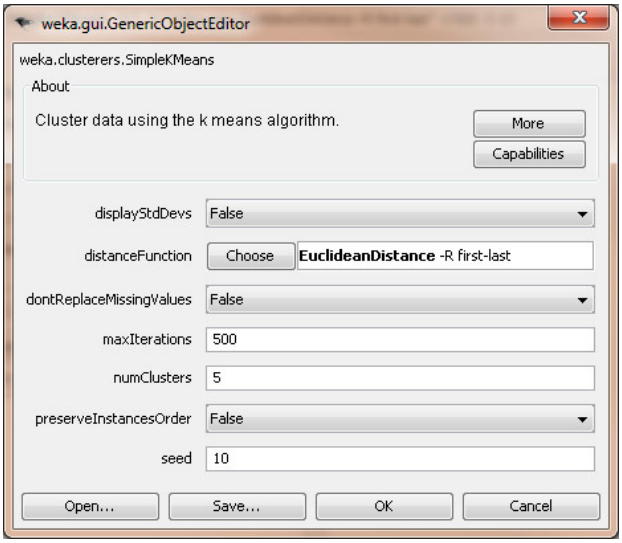


Рис. 2.7 – Налаштування алгоритму кластеризації

Тепер ми можемо приступити до створення моделі. Як вже зазначалося вище, для розбиття 100 рядків на 5 кластерів за допомогою електронних таблиць було б потрібно декілька годин, проте WEKA видає нам результат менш ніж за секунду. Результат обробки наших даних показаний нижче.

Результати кластеризації

```
                          Cluster#
Attribute       Full Data      0       1       2       3       4
                  (100)       (26)    (27)    (5)     (14)    (28)
=============================================================================
Dealership          0.6     0.9615  0.6667       1   0.8571       0
Showroom           0.72     0.6923  0.6667       0   0.5714       1
ComputerSearch     0.43     0.6538       0       1   0.8571  0.3214
M5                 0.53     0.4615   0.963       1   0.7143       0
3Series            0.55     0.3846  0.4444     0.8   0.0714       1
Z4                 0.45     0.5385       0     0.8   0.5714  0.6786
Financing          0.61     0.4615  0.6296     0.8        1     0.5
```

Purchase 0.39 0 0.5185 0.4 1 0.3214

Clustered Instances

0 26 (26%)
1 27 (27%)
2 5 (5%)
3 14 (14%)
4 28 (28%)

Як нам інтерпретувати отриманий результат? Дані кластеризації показують, яким чином сформований кожен кластер: значення «1» означає, що у всіх даних в цьому кластері відповідний атрибут дорівнює 1, а значення «0» означає, що у всіх даних в цьому кластері відповідний атрибут дорівнює 0. Дані відповідають середньому значенню атрибута у кластері. Кожен кластер характеризує певний тип поведінки клієнтів, таким чином, на підставі нашого розбиття ми можемо зробити деякі корисні висновки:

Кластер 0 - цю групу відвідувачів можна було б назвати «мрійники». Вони бродять навколо дилерського центру, розглядаючи машини, виставлені на зовнішній парковці, але ніколи не заходять всередину, і, гірше того, ніколи нічого не купують.

Кластер 1 - цю групу слід було б назвати «шанувальники M5», оскільки вони відразу ж підходять до виставлених автомобілів цієї моделі, повністю ігноруючи BMW серії 3 або Z4. Тим не менш, ця група не відрізняється високими показниками покупки машин - всього 52%. Це потенційно може свідчити про недостатньо продуману стратегію продажів і про необхідність поліпшити роботу дилерського центру, наприклад, за рахунок збільшення кількості продавців у секції M5.

Кластер 2 - ця група настільки мала, що ми могли б назвати її вибраковуванням. Справа в тому, що дані цієї групи статистично досить розкидані, і ми не можемо зробити будь-яких певних висновків щодо поведінки відвідувачів, що потрапили в цей кластер (подібна ситуація може вказувати на те, що вам слід скоротити кількість кластерів в моделі)

Кластер 3 - цю групу слід було б назвати «улюбленці BMW», тому що відвідувачі, що потрапили в цей кластер, завжди купують машину і отримують необхідне фінансування. Зверніть увагу, дані цього кластеру демонструють цікаву модель поведінки цих покупців: спочатку вони оглядають виставлені на парковці машини, а потім звертаються до пошукової системи дилерського центру. Як правило, вони купують моделі M5 або Z4, але ніколи не беруть моделі третьої серії. Дані цього кластеру вказують на те, що дилерському центру слід активніше привертати увагу до пошукових комп'ютерів (можливо, винести їх на зовнішню парковку), і крім того, слід знайти якийсь спосіб виділити моделі M5 і Z4 в результатах пошуку, щоб гарантовано звернути на них увагу відвідувачів. Після того, як відвідувач, що потрапив в цей кластер, вибрав певну модель автомобіля, він гарантовано отримує необхідний кредит і здійснює покупку.

Кластер 4 - цю групу можна назвати «початківці власники BMW», оскільки вони завжди шукають моделі 3 серії і ніколи не цікавляться більш дорогими M5. Вони відразу ж проходять в демонстраційний зал, не витрачаючи час на огляд машин на зовнішній стоянці. Крім того, вони не користуються пошуковою системою центру. Приблизно 50% цієї групи отримують схвалення по кредиту, тим не менш, покупку роблять всього 32% учасників. Аналізуючи дані цього кластеру, можна зробити наступний висновок: відвідувачі цієї групи хотіли б купити свій перший BMW і точно знають, яка машина їм потрібна (модель 3 серії з мінімальною конфігурацією). Однак, для того щоб купити машину, їм потрібно отримати позитивне рішення по кредиту. Щоб підвищити рівень продажів серед відвідувачів 4 кластера, дилерському центру слід було б знизити рівень вимог для отримання кредиту або знизити ціни на моделі 3 серії.

Ще один цікавий спосіб вивчення результатів кластеризації - це візуальне подання даних. Клацніть правою кнопкою мишки в секції **Result List** закладки **Cluster**. У контекстному меню виберіть опцію **Visualize Cluster Assignments**. В результаті відкриється вікно з графічним представленням результатів кластеризації, налаштування якого ви можете вибрати найбільш

зручним для вас чином. Для нашого прикладу, змініть настройку осі **X** так, щоб вона відповідала кількості автомобілів M5 (**M5 (Num)**), а настройку осі **Y** - так, щоб вона показувала кількість куплених автомобілів (**Purchase (Num)**), і вкажіть виділення кожного кластера окремим кольором (для цього встановіть значення поля **Color** в **Cluster (Nom)**). Такі налаштування допоможуть нам оцінити розподіл по кластерах залежно від того, скільки людина цікавилося BMW M5, і скільки чоловік купило цю модель. Крім того, посуньте покажчик **Jitter** приблизно на три чверті у бік максимуму, це штучно збільшить розкид між групами точок, щоб нам було зручніше їх переглядати.

Чи відповідає візуальне відображення кластеризації тим висновкам, які ми зробили на підставі даних в одержаному результаті кластеризації? Як ми бачимо, поблизу точки **X = 1**, **Y = 1** (відвідувачі, які цікавилися автомобілями моделі M5 і купили їх) розташовані тільки два кластери: **1** і **3**. Аналогічно, поблизу точки **X = 0**, **Y = 0** розташовані тільки два кластери: **4** і **0**. Чи відповідає це нашим висновкам? Так, відповідає. Кластери **1** і **3** купують BMW M5, в той час як кластер **0** не купує нічого, а кластер **4** шукає BMW серії 3. На рис. 2.8 показано візуальне відображення кластерів нашої моделі. Пропонуємо вам самостійно попрактикуватися у виявленні інших трендів і течій, змінюючи налаштування осей **X** і **Y**.

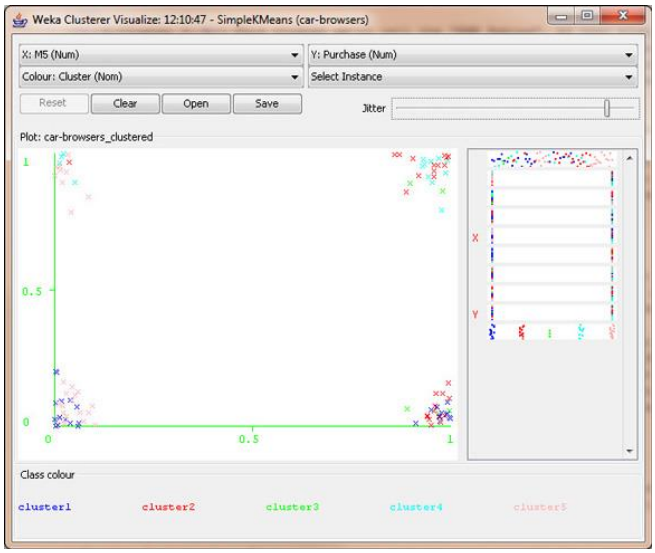


Рис. 2.8 – Візуальне відображення кластеризації

Висновки:

У даній лабораторній роботі:

- **Waikato Environment for Knowledge Analysis (WEKA)**, що дозволяє аналізувати значні обсяги інформації та визначати існуючі в них тренди і залежності;
- Ознайомитись з методами класифікації та кластеризації;
- Ознайомитись з деревом рішень;

2.4 Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Текст розробленого програмного забезпечення з коментарями, а також текст програми для тестування розроблених генетичних методів.
4. Графіки та аналітичні вирази обраних тестових функцій.
5. Результати роботи програмного забезпечення.
6. Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх аналіз

2.5 Контрольні питання

1. З якими методами інтелектуального аналізу даних ми ознайомилися в даній лабораторній роботі?
2. Під якими іншими назвами відомий метод класифікації?
3. У чому полягає основна перевага класифікаційних дерев?
4. Суть підходу створення моделі на основі навчальної послідовності (training set).
5. З якою метою в методі класифікаційного аналізу набір відомих даних ділиться на дві частини?

6. Принцип відсікання гілок. Для чого може бути потрібно видаляти інформацію з дерева рішень?
7. Сенс проблеми помилкового розпізнавання. Що є більш небажаним: хибно-позитивне чи хибно-негативне розпізнавання? Яке співвідношення хибно-негативних розпізнавань до хибно-позитивних може вважатися прийнятним?
8. В результаті аналізу для тестового і навчального набору даних одержана приблизно однакова точність. Як це вплине на нові дані які будуть використовуватися в цій моделі в майбутньому?
9. Коли використовується метод кластерного аналізу?
10. Основний недолік методу кластеризації.

3 ЛАБОРАТОРНА РАБОТА №3.АСОЦІАТИВНІ ПРАВИЛА. АЛГОРИТМ APRIORI.

3.1 Мета роботи

Використовуючи пакет Matlab одержати навички розробки асоціативних правил. Вивчити алгоритм Apriori. Побудувати асоціативні правила при різних мірах підтримки і достовірності. Проаналізувати отримані результати. Побудувати граф отриманих правил, проаналізувати результат. Побудувати матрицю частот. Побудувати матрицю підтримки правил, проаналізувати результат.

3.2 Методичні рекомендації до самостійної роботи студентів

Однією з задач Data Mining є асоціація. Метою пошуку асоціативних правил (association rule) є знаходження закономірностей між зв'язаними подіями в базах даних.

Дуже часто покупці придбавають не один товар, а декілька. В більшості випадків між цими товарами існує взаємозв'язок. Так, наприклад, покупець, що придбаває макаронні вироби, швидше за все, схоче придбати також кетчуп. Ця інформація може бути використана для розміщення товару на полицях крамниці.

Наведемо простий приклад асоціативного правила: покупець, що придбаває банку фарби, придбає пензлик для фарби з вірогідністю 50%.

Вперше ця задача була запропонована пошуку асоціативних правил для знаходження типових шаблонів покупок, які придбають в супермаркетах, тому іноді її ще називають аналізом ринкової корзини (market basket analysis).

Хай є база даних, що складається з купівельних транзакцій. Кожна транзакція – це набір товарів, куплених покупцем за один візит. Таку транзакцію ще називають ринковою корзиною.

Визначення 1. Хай I = {i_1, i_2, i3 ... i_n} – безліч (набір) товарів, званих елементами. Хай D - безліч транзакцій, де кожна транзакція T – це набір елементів з I, $T \subseteq I$. Кожна транзакція є бінарним вектором, де t[k]=1, якщо i_k елемент присутній в транзакції, інакше t[k]=0. Ми говоримо, що транзакція T містить X, деякий набір елементів з I, якщо $X \subset T$. Асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \cap Y = \otimes$. Правило $X \Rightarrow Y$ має підтримку s (support), якщо s% транзакцій з D, містять $X \cup Y$, supp(X $\Rightarrow$ Y) = supp($X \cup Y$). Достовірність правила показує, яка вірогідність того, що з X слідує У. Правило $X \Rightarrow Y$ справедливо з достовірністю (confidence) з, якщо c% транзакцій з D, що містять X, також містять У, conf($X \Rightarrow$ Y) = supp($X \cup Y$)/supp(X).

Покажемо на конкретному прикладі: "75% транзакцій, що містять хліб, також містять молоко. 3% від загального числа всіх транзакцій містять обидва товари". 75% - це достовірність (confidence) правила, 3% це підтримка (support), або "Хліб" "Молоко" з вірогідністю 75%.

Іншими словами, метою аналізу є встановлення наступної залежності: якщо в транзакції зустрівся деякий набір елементів X, то на підставі цього можна зробити висновок про те, що інший набір елементів У також повинен з'явитись в цій транзакції. Встановлення такої залежності дає нам можливість знаходити дуже прості і інтуїтивно зрозумілі правила.

Алгоритми пошуку асоціативних правил призначені для знаходження всіх правил X У, причому підтримка і достовірність цих правил повинна бути вищою за деякі наперед певні пороги, звані відповідно мінімальною підтримкою (minsupport) і мінімальною достовірністю (minconfidence).

Задача знаходження асоціативних правил розбивається на дві підзадачі:

а) знаходження всіх наборів елементів, які задовольняють порогу minsupport. Такі набори елементів називаються тими, що часто зустрічаються;

б) генерація правил з наборів елементів, знайдених згідно п. а) з достовірністю, що задовольняє порогу minconfidence.

Один з перших алгоритмів, ефективно вирішальних подібний клас задач, – це алгоритм APriori. Окрім цього алгоритму останнім часом був розроблений ряд інших алгоритмів: DHP, Partition, DIC і інші.

Значення для параметрів мінімальна підтримка і мінімальна достовірність вибираються так, щоб обмежити кількість знайдених правил. Якщо підтримка має велике значення, то алгоритми будуть знаходити правила, добре відомі аналітикам або настільки очевидні, що немає ніякого значення проводити такий аналіз. З другого боку, низьке значення підтримки веде до генерації величезної кількості правил, що, звичайно, вимагає істотних обчислювальних ресурсів. Проте, більшість цікавих правил знаходиться саме при низькому значенні порогу підтримки. Хоча дуже низьке значення підтримки веде до генерації статистично необґрунтованих правил.

Пошук асоціативних правил зовсім не тривіальна задача, як може здатися на перший погляд. Одна з проблем – алгоритмічна складність при знаходженні наборів елементів, які часто зустрічаються, оскільки із зростанням числа елементів експоненційно росте число потенційних наборів елементів.

Узагальнені асоціативні правила (Generalized Association Rules)

При пошуку асоціативних правил, ми припускали, що всі аналізовані елементи однорідні. Повертаючись до аналізу ринкової корзини, це товари, абсолютно однакові атрибути, що мають, за винятком назви. Проте не складе великих труднощів доповнити транзакцію інформацією про те, до якої товарної групи входить товар і побудувати ієрархію товарів.

Наприклад, якщо Покупець купив товар з групи "Безалкогольні напої", то він купить і товар з групи "Молочні продукти" або "Сік". Ці правила носять назву узагальнених асоціативних правил.

Визначення 2. Узагальненим асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \cap Y = \otimes$ і де жоден з елементів, що входять в набір У, не є предком жодного елемента, що входить в X. Підтримка і

достовірність підраховуються так само, як і у разі асоціативних правил (див. Визначення 1).

Введення додаткової інформації про угрупування елементів у вигляді ієрархії дасть наступні переваги:

а) це допомагає встановити асоціативні правила не тільки між окремими елементами, але і між різними рівнями ієрархії (групами);
б) окремі елементи можуть мати недостатню підтримку, але в цілому група може задовольняти порогу minsupport;
в) для знаходження таких правил можна використовувати будь-який з вищеназваних алгоритмів.

Групувати елементи можна не тільки по входженню в певну товарну групу, але і по інших характеристиках, наприклад за ціною (дешево, дорого), бренду і т.д.

Алгоритм пошуку асоціативних правил, заснований на аналізі частих наукових наборів. Спочатку в базі даних транзакцій шукаються усі частини наукових наборів, а потім генеруються асоціативні правила на основі тих з них, які задовольняють заданому рівню підтримки і достовірності.

При цьому для скорочення простору пошуку асоціативних правил використовується властивість апріорності. Воно затверджує, що якщо науковий набір Z не є частим, то додавання будь – якого нового предмету A до набору Z не робить його таким. Іншими словами, якщо набір Z не є частим, то і Z + A – теж.

Властивість анти–монотонності

Виявлення наборів елементів, що часто зустрічаються, – операція, що вимагає багато обчислювальних ресурсів і, відповідно, часу. Примітивний підхід до рішення даної задачі – простий перебір всіх можливих наборів елементів. Це потребує $O(2^{|I|})$ операцій, де |I| – кількість елементів. Apriori використовує одну з властивостей підтримки, що свідчить: підтримка будь-якого набору елементів не

може перевищувати мінімальної підтримки будь-якої з його підмножин. Наприклад, підтримка 3–елементного набору {Хліб, Масло, Молоко} буде завжди менше або рівно підтримці 2–елементних наборів {Хліб, Масло}, {Хліб, Молоко}$\Rightarrow${Масло, Молоко}. Річ у тому, що будь-яка транзакція, що містить {Хліб, Масло, Молоко}, також повинна містити {Хліб, Масло}, {Хліб, Молоко}, {Масло, Молоко}, причому зворотне не вірно.

Ця властивість носить назву анти–монотонності і служить для зниження розмірності простору пошуку. Не май ми в наявності такої властивості, знаходження багатоелементних наборів було б практично нездійсненною задачею у зв'язку з експоненціальним зростанням обчислень.

Властивості анти–монотонності можна дати і інше формулювання: із зростанням розміру набору елементів підтримка зменшується, або залишається такою ж. Зі всього, що було сказано раніше витікає, що будь–який k–елементний набір буде часто зустрічатися тоді і тільки тоді, коли всі його (k–1)–елементні підмножини будуть часто зустрічатись. Всі можливі набори елементів з I можна представити у вигляді грат, що починаються з порожньої множини, потім на 1 рівні 1–елементні набори, на 2–м – 2–елементні і т.д. На k рівні представлені k–елементні набори, пов'язані зі всіма своїми (k – 1) – елементними підмножинами.

Розглянемо рис. 3.1, ілюструючи набір елементів I – {A, B, 3, D}. Припустимо, що набір з елементів {A, B} має підтримку нижче заданого порогу і, відповідно, не є тим, що часто зустрічається. Тоді, згідно властивості анти–монотонності, всі його супермножини також не є тими, що часто зустрічаються і відкидаються. Вся ця гілка, починаючи з {A, B}, виділена фоном. Використовування цієї евристики дозволяє істотно скоротити простір пошуку.

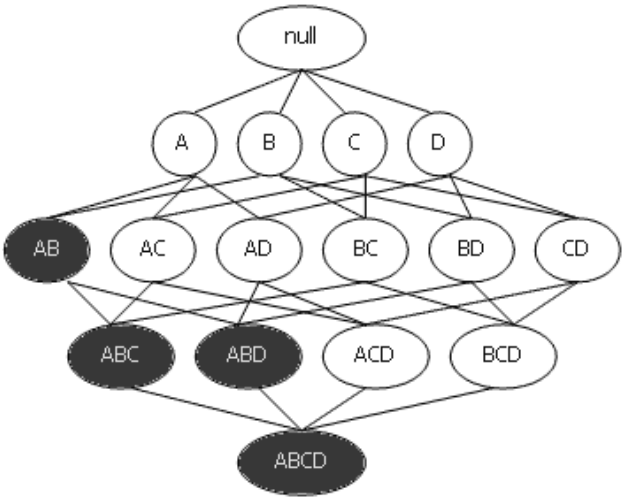


Рисунок 3.1 – Набір елементів I

Алгоритм Apriori

Для того, щоб було можливе застосувати алгоритм, необхідно провести предобработку даних: по-перше, привести всі дані до бінарного вигляду; по-друге, змінити структуру даних[9]. Вигляд транзакційної бази даних представлен в таблиці 3.1 і таблиці 3.2.

Таблиця 3.1 – Звичайний вигляд

Номер транзакції	**Назва елемента**	**Кількість**
1001	A	2
1001	D	3
1001	E	1
1002	A	2
1002	F	1
1003	B	2
1003	A	2

1003	C	2
…	…	…

Таблиця 3.2 – Нормалізований вигляд

TID	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**K**	**…**
1001	1	0	0	1	1	0	0	0	0	0	…
1002	1	0	0	0	0	1	0	0	0	0	…
1003	1	1	1	0	0	0	0	0	1	0	…

Кількість стовпців в таблиці рівно кількості елементів, присутніх в безлічі транзакцій D. Кожний запис відповідає транзакції, де у відповідному стовпці стоїть 1, якщо елемент присутній в транзакції, і 0 в осоружному випадку. Помітимо, що початковий вид таблиці може бути відмінним від приведеного в таблиці 2.1. Головне, щоб дані були перетворені до нормалізованого вигляду, інакше алгоритм не застосовний. Крім того, всі елементи повинні бути впорядкований в алфавітному порядку (якщо це числа, вони повинні бути впорядкований в числовому порядку).

На першому кроці алгоритму підраховуються 1–елементні набори, що часто зустрічаються. Для цього необхідно пройтися по всьому набору даних і підрахувати для них підтримку, тобто скільки разів зустрічається в базі.

Наступні кроки складатимуться з двох частин: генерації наборів елементів (їх називають кандидатами) і підрахунку підтримки, що потенційно часто зустрічаються, для кандидатів[4].

Описаний вище алгоритм можна записати у вигляді наступного псевдокоду:

F_1 = {1-елементні набори, що часто зустрічаються}
для (k=2; F_{k-1} <> ∅; k++) {
C_k = Apriorigen (F_{k-1}) // генерація кандидатів
для всіх транзакцій t ∈T {

C_t = subset(C_k, t)// видалення надмірних правил
для всіх кандидатів c $\in C_t$
c.count ++
}
F_k = {c $\in C_k$ | c.count >= minsupport} // відбір кандидатів
}
Результат $\cup F_k$

Опишемо функцію генерації кандидатів. На цей раз немає ніякої необхідності знов звертатися до бази даних. Для того, щоб отримати k–елементні набори, скористаємося (k–1)–елементними наборами, які були визначені на попередньому кроці і є тими, що часто зустрічаються.

Пригадаємо, що наш початковий набір зберігається у впорядкованому вигляді.

Генерація кандидатів також складатиметься з двох кроків.

Крок 1. Об'єднання. Кожний кандидат $C_к$ формуватиметься шляхом розширення набору розміру (k–1), що часто зустрічається, додаванням елемента з іншого (k–1)–елементного набору.

Приведемо алгоритм цієї функції Apriorigen у вигляді невеликого SQL – подібного запиту.

insert into C_k
select p.item$_1$, p.item$_2$., p.item$_{k-1}$, q.item$_{k-1}$
From F_{k-1} p, F_{k-1} q
where p.item$_1$= q.item$_1$, p.item$_2$ = q.item$_2$., p.item$_{k-2}$ = q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$

Крок 2. Видалення надмірних правил. На підставі властивості анти–монотонності, слід видалити всі набори c $\in C_k$ якщо хоча б одна з його (k–1) підмножин не є тим, що часто зустрічається.

Після генерації кандидатів наступною задачею є підрахунок підтримки для кожного кандидата. Очевидно, що кількість кандидатів може бути дуже великою і потрібен ефективний спосіб підрахунку. Найтривіальніший спосіб – порівняти кожну транзакцію з кожним кандидатом. Але це далеко не краще рішення.

Набагато швидше і ефективно використовувати підхід, заснований на зберіганні кандидатів в хеш–дереві. Внутрішні вузли дерева містять хеш–таблиці з покажчиками на нащадків, а листя – на кандидатів. Це дерево нам стане в нагоді для швидкого підрахунку підтримки для кандидатів.

Хеш–дерево будується кожного разу, коли формуються кандидати. Спочатку дерево складається тільки з кореня, який є листом, і не містить ніяких кандидатів – наборів. Кожного разу коли формується новий кандидат, він заноситься в корінь дерева і так до тих пір, поки кількість кандидатів в корені – листі не перевищить якогось порогу. Як тільки кількість кандидатів стає більше порогу, корінь перетвориться в хеш–таблицю, тобто стає внутрішнім вузлом, і для нього створюються нащадки – листя. І всі приклади розподіляються по вузлах – нащадкам згідно з хеш–значеннями елементів, що входять в набір, і т.п. Кожний новий кандидат хеширується на внутрішніх вузлах, поки він не досягне першого вузла–листа, де він і зберігатиметься, поки кількість наборів знову ж таки не перевищить порогу.

Хеш–дерево з кандидатами–наборами побудовано, зараз, використовуючи хеш–дерево, легко підрахувати підтримку для кожного кандидата. Для цього потрібно "пропустити" кожну транзакцію через дерево і збільшити лічильники для тих кандидатів, чиї елементи також містяться і в транзакції, тобто $C_k \cap T_i = C_k$. На кореневому рівні хеш–функція застосовується до кожного елемента з транзакції. Далі, на другому рівні, хеш–функція застосовується до других елементів і т.д. На k–рівні хеширується k–елемент. І так до тих пір, поки не досягнемо листа. Якщо кандидат, що зберігається в листі, є підмножиною даної транзакції, тоді збільшуємо лічильник підтримки цього кандидата на одиницю.

Після того, як кожна транзакція з початкового набору даних "пропущена" через дерево, можна перевірити чи задовольняють значення підтримки кандидатів мінімальному порогу. Кандидати, для яких ця умова виконується, переносяться в розряд часто зустрічаємих. Крім того, слід запам'ятати і підтримку набору, вона нам стане в нагоді при витяганні правил. Ці ж дії застосовуються для знаходження (k+1)–елементних наборів і т.д.

Після того, як знайдені всі часто зустрічаємі набори елементів можна приступити безпосередньо до генерації правил.

Витягання правил – менш трудомістка задача. По–перше, для підрахунку достовірності правила достатньо знати підтримку самого набору і множини, що лежить в умові правила. Наприклад, є набір {A, B, C}, що часто зустрічається, і потрібно підрахувати достовірність для правила AB⇒C. Підтримка самого набору нам відома, але і його множина {A, B}, що лежить в умові правила, також часто зустрічається через властивість анти–монотонності, і значить, його підтримка нам відома. Тоді ми легко зможемо підрахувати достовірність. Це позбавляє нас від небажаного перегляду бази транзакцій, який був б потрібен в тому випадку, якщо б це підтримка була невідома.

Щоб витягнути правило із часто зустрічає мого набору F, слід знайти всі його не порожні підмножини. І для кожної підмножини s ми зможемо сформулювати правило s ⇒(F – s), якщо достовірність правила conf(s ⇒(F – s)) = supp(F)/supp(s) не менше порогу minconf.

Помітимо, що чисельник залишається постійним. Тоді достовірність має мінімальне значення, якщо знаменник має максимальне значення, а це відбувається у тому випадку, коли в умові правила є набір, що складається з одного елемента. Все супермножина даної множини має меншу або рівну підтримку і, відповідно, більше значення достовірності. Ця властивість може бути використана при витяганні правил. Якщо ми почнемо витягувати правила, розглядаючи спочатку тільки один елемент в умові правила, і це правило має необхідну підтримку, тоді всі правила, де в умові стоїть супермножина цього елемента, також мають значення достовірності вище заданого порогу. Наприклад, якщо правило A⇒BCDE задовольняє мінімальному порогу достовірності minconf, тоді AB⇒CDE також задовольняє. Для того, щоб витягнути всі правила використовується рекурсивна процедура. Важливе зауваження: будь-яке правило, складене з набору, що часто зустрічається, повинне містити всі елементи набору. Наприклад, якщо набір складається з елементів {A, B, C}, то правило A⇒B не повинне розглядатися.

3.3 Порядок виконання роботи

1. Ознайомитися з конспектом лекцій та рекомендованою літературою за темою роботи, а також з додатком А, що містить опис програмного забезпечення для видобування асоціативних правил з великих масивів даних.

2. Сформувати набір даних для обробки та аналізу.

3. Розробити за допомогою середовища Matlab програмне забезпечення для видобування асоціативних правил з великих масивів даних або вивчити рекомендоване програмне забезпечення (пакет Armada модулю Matlab) та здійснити обробку набору даних з метою виділення асоціативних правил.

Асоціативні правила у пакеті matlab

Операції з асоціативними правилами у пакеті MATLAB дозволяє виконувати модуль *ARMADA*. Він дозволяє створювати асоціативні правила по заданим користувачем даним в рамках середовища MATLAB.

Для запуску модулю *ARMADA* необхідно зробити папку, де знаходиться цей модулю, поточною, після чого в командному рядку середовища MATLAB написати наступну команду: armada

В результаті з’явиться головна інтерфейсна форма модулю, в якій необхідно обрати файл з вхідними даними та параметри для аналізу даних (рис. 3.2).

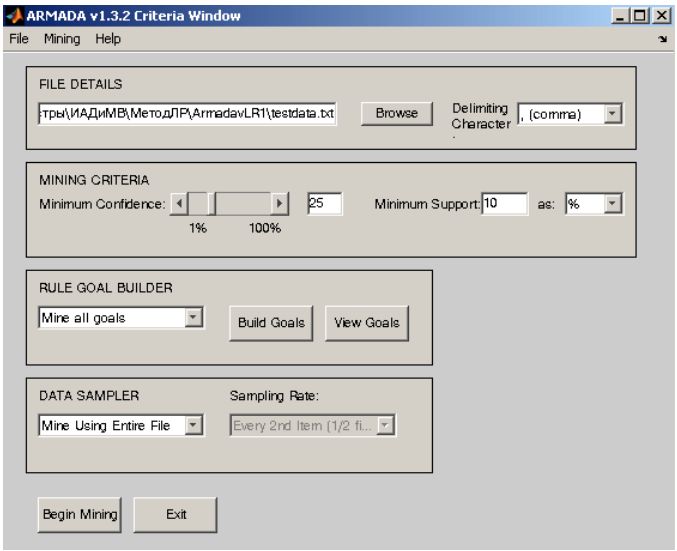


Рисунок 3.2 – Головна інтерфейсна форма модулю *ARMADA*

Для інтелектуального аналізу даних за допомогою асоціативних правил в полі FILE DETAILS необхідно ввести шлях до файлу з даними для аналізу. Файл з даними для аналізу можна також обрати у віконному режимі, натиснувши кнопку Browse. В полі зі списком Delimiting Character необхідно вказати символ, що відокремлює дані одне від одного.

Компоненти головної форми Minimum Confidence Minimum Support призначені для введення параметрів minsupport та minconfidence, відповідно.

Для виконання аналізу даних та отримання вихідної інформації у вигляді асоціативних правил необхідно натиснути кнопку Begin Mining, після чого відбудеться процес аналізу даних, та з'явиться форма з результатами роботи програми (рис. 3.3). Вихід з програми виконується за допомогою кнопки Exit.

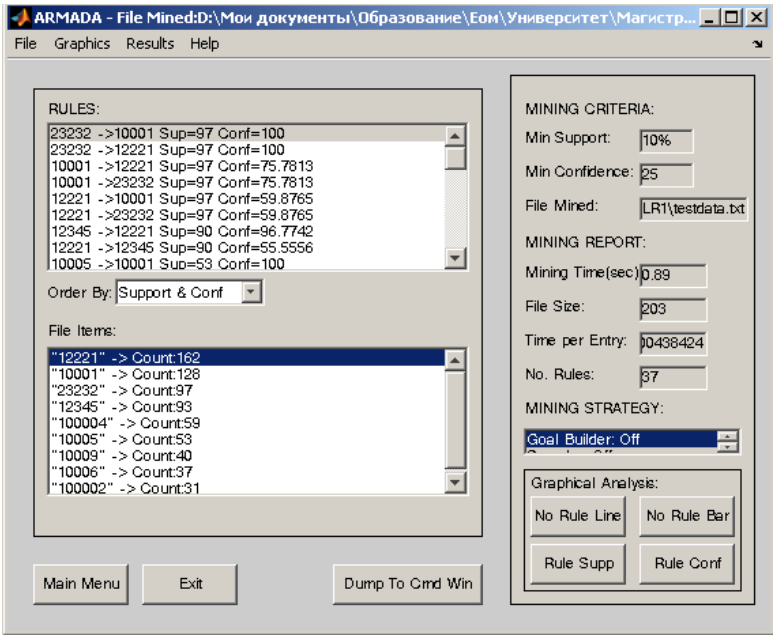


Рисунок 3.3 – Результати аналізу даних за допомогою модулю *ARMADA*

В області RULES наведено отримані асоціативні правила, в області File Items – елементи файлу для аналізу та кількість разів, які вони зустрічаються у файлі.

Компоненти в області MINING CRITERIA відображають параметри аналізу: значення параметрів minsupport та minconfidence, а також файл із даними для аналізу.

В області MINING REPORT відображається інформація про хід виконання аналізу:

– час, витрачений на виконання аналізу (Mining Time);

– розмір файлу (File Size);

– час, затрачений на один запис (Time per Entry);

– кількість отриманих асоціативних правил (No. Rules)

За допомогою головного меню отримані дані можна зберегти на диск (File –> Save) та отримати графіки, що відображають процес аналізу даних (Graphics).

4. Оформити звіт з роботи.

5. Відповісти на контрольні питання.

Висновки:

У даній лабораторній роботі:

- Розглянуто пакет WEKA, сформовано здачачі;

3.4 Зміст звіту

1. Тема та мета роботи.

2. Короткі теоретичні відомості.

3. Набір даних для обробки (якщо він великий – навести фрагмент).

4. Детальний опис процесу використання програмного забезпечення для обробки набору даних, що має бути проілюстрований зображеннями екранних форм з описом кожного їх елементу та коментарями до кожної дії.

5. Текст програми з коментарями.

6. Результати роботи програмного забезпечення (набір отриманих правил, інші характеристики).

7. Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

3.5 Контрольні питання

1. Що таке асоціативне правило?
2. Для чого призначені асоціативні правила?
3. Дати визначення понять підтримки та достовірності правила.
4. Яке призначення алгоритмів пошуку асоціативних правил?
5. На які підзадачі розбивається задача знаходження асоціативних правил?
6. Які методи використовуються для знаходження асоціативних правил?
7. Яким чином обираються значення параметрів minsupport та minconfidence?
8. Що таке числові асоціативні правила?
9. Поясніть поняття “узагальнене асоціативне правило”.
10. Що називається ієрархією елементів?
11. Які переваги дає введення додаткової інформації про групування елементів?
12. Поясніть, які проблеми можуть виникнути при безпосередньому застосуванні алгоритмів знаходження асоціативних правил.
13. В чому полягає сутність виявлення узагальнених асоціативних правил?
14. Яким чином визначають “цікаві” правила? В чому полягає актуальність такого процесу?
15. Дати визначення понять батьківського правила (предка) та найближчого батьківського правила.
16. Порівняйте поняття цікавого та частково цікавого правила.
17. Які проблеми усуває алгоритм обчислення узагальнених асоціативних правил?
18. З яких етапів складається процес обчислення узагальнених асоціативних правил?
19. Проаналізуйте базовий алгоритм пошуку множин, що зустрічаються часто.
20. Опишіть алгоритм генерації кандидатів.
21. Яким чином використовується хеш-дерево для підрахунку підтримки кандидатів? Як відбувається процес побудови такого дерева?

22. Виконайте порівняльний аналіз базового та покращеного алгоритмів пошуку множин, що зустрічаються часто.
23. За рахунок яких оптимізацій відбувається покращення базового алгоритму пошуку множин, що зустрічаються часто?
24. В чому полягає сутність масштабуємого алгоритму пошуку асоціативних правил Apriori?
25. Яким чином перетворюються дані для можливості використання алгоритму Apriori?
26. Яка властивість використовується в алгоритмі Apriori? Для чого вона використовується?
27. Наведіть послідовність виконання алгоритму Apriori.
28. Опишіть функцію генерації кандидатів в алгоритмі Apriori.
29. Як відбувається підрахунок підтримки для кожного кандидату в алгоритмі Apriori? Для чого в цій процедурі використовують хеш-дерево?
30. Як здійснити добування правил з набору, що часто зустрічається?
31. Проаналізуйте внутрішню структуру модулю Armada пакету Matlab: основні змінні, параметри, методи та функції, їх призначення та використання.

4 ЛАБОРАТОРНА РАБОТА №4. ГЕНЕТИЧНІ АЛГОРИТМИ.

4.1 Мета роботи

Вивчити основні методи генетичного пошуку.Навчитися використовувати генетичні методи для розв’язку оптимізаційних задач.

4.2 Методичні рекомендації до самостійної роботи студентів

Генетичні успадкування – концептуальні засади генетичних алгоритмів

У загальному значенні *генетичні алгоритми (Genetic Algorithms)* — це тип алгоритмів, інспірованих механізмами еволюції живої природи, які застосовуються, головно, до задач глобальної оптимізації (зокрема, задач комбінаторної оптимізації) і деякою мірою для дейтамайнінгу, зокрема, для комбінування шаблонів з правил індукції, які були відкриті до цього, навчання нейромереж, пошуку зразків у даних, відкриття шаблонів у тексті тощо. Генетичні алгоритми належать нині до стандартного інструментарію методів дейтамайнінгу.

Ідея генетичних алгоритмів запозичена з живої природи і полягає в машинній організації еволюційного процесу створення, модифікації і відбору кращих розв'язків, виходячи з того, що в процесі відтворення і модифікації розв'язків кращі з них (подібно До процесу селекції в рослинництві й тваринництві) можуть дати ще ліпших «нащадків», тобто нові, прийнятніші варіанти розв'язання задачі. Щоб краще зрозуміти концептуальні засади генетичних алгоритмів, зупинимося на короткому огляді механізмів природного добору і генетичного успадкування, що розглядаються в еволюційній теорії зародження і розвитку життя на нашій планеті. Ця теорія стверджує, що кожний біологічний вид ціле спрямовано розвивається й змінюється так, щоб у найкращий спосіб пристосуватися до навколишнього середовища.

Ключову роль в еволюції відіграє природний добір. Його суть полягає в тому, що найпристосованіші особи краще виживають і приносять більше потомства, ніж менш пристосовані. При цьому завдяки передаванню генетичної інформації, що називається *генетичним успадковуванням,* нащадки успадковують від батьків основні властивості. Проте слід зауважити, що сам по собі природний добір ще не забезпечує розвитку біологічного виду. Дійсно, якщо передбачити, що всі нащадки народжуються приблизно однаковими, то покоління будуть відрізнятися тільки за чисельністю, але не за пристосованістю. Тому дуже важливо вивчити, у який спосіб відбувається успадкування, тобто як властивості нащадка залежать від властивостей батьків.

Майже в кожній клітині будь-якої тварини є ряд хромосом, що несуть інформацію про цю тварину. Основна частина хромосоми — нитка ДНК (молекула дезоксирибоза Нуклеїнової Кислоти), яка складається з чотирьох видів спеціальних з'єднань (молекул) — нуклеотидів, що чергуються в певній послідовності. Нуклеотиди позначають буквами A, T, C і G, і саме порядок їх розміщеня є кодом усіх генетичних властивостей даного організму. Кажучи точніше, ДНК визначає, які хімічні реакції будуть відбуватися в даній клітині, як вона буде розвиватися і які функції виконуватиме. Отже, генетичний код окремого індивідуума — це просто дуже довгий рядок комбінацій із чотирьох букв A, T, C і G, а сам ген — це відрізок ланцюга ДНК, що відповідає за певну властивість особи, наприклад за колір очей, тип волосся, колір шкіри і т. д. Різні значення генів називають *аллелями.* Вся сукупність генетичних ознак людини кодується за допомогою приблизно 60 тис. генів, які разом містять більше ніж 90 млн нуклеотидів.

У мейозі, зокрема, відбувається наступне: парні хромосоми соматичної клітини зближуються впритул, потім їх нитки ДНК розриваються в кількох випадкових місцях і хромосоми обмінюються своїми ідентичними ділянками. Цей процес забезпечує появу нових варіантів хромосом І називається *перехрещуванням хромосом* або *кросинговером* (від анг. *crossing-over).* Кожна з

хромосом, що знову з'явилася, виявиться потім усередині однієї зі статевих клітин, і її генетична інформація може реалізуватися в нащадках даної особи.

Другим важливим чинником, що впливає на спадковість, ***£ мутації,*** тобто раптові спадкові зміни організму або його частин, ознак, властивостей, які виражаються у зміні деяких дільниць ДНК. Мутації також випадкові і можуть бути викликані різними зовнішніми чинниками, такими, наприклад, як радіоактивне опромінення. Якщо мутація сталася в статевій клітині, то змінений ген може передатися нащадку й виявитися у вигляді спадкової хвороби або в інших нових властивостях нащадка. Вважається, що саме мутації є причиною появи нових біологічних видів, а кросинговер визначає мінливість уже всередині виду (наприклад, генетичні відмінності між людьми).

Важливе місце в еволюційній теорії відводиться поняттю *популяції* як елементарній еволюційній одиниці. ***Популяція*** — це сукупність особин певного виду організмів, які здатні до вільного схрещування, населяють певну територію і деякою мірою ізольовані від сусідніх популяцій. У рамках кожної популяції відбувається процес розмноження — *репродукції (Reproduction),* що являє собою комбінацію послідовностей (strings, хромосом) у опуляци для створення нової послідовності (нащадка). За реродукції нащадок бере частини позицій генів від обох батьків, матиме частину ознак кожного із них. На рис. 9.13а) показана спрощена схема процесу репродукції, де ознаки батьків виражені хромосомою, котра складається з шести генів, що мають дві аллелі, позначені на схемі нулями і одиницями. Нащадок отримав чотири гени від другого батька (перша, друга, третя і шоста позиція) і два від першого (четверта і п'ята позиції).

У генетичних алгоритмах важливе значення мають: формування початкового ряду елементів (популяції), операції кросинговера, що в теорії генетичних алгоритмів частіше називають *кросовером (Cross-over),* і *мутації (Mutation).*

Кросовер -— це комбінування (змішування) хромосом шляхом замін значень генів і утворення нових хромосом на їх місцях. На рис. 1. *б)* наведена

спрощена схема кросовера, де показано, як шляхом заміни ідентичних ділянок двох батьків отримані два нащадки з новими ознаками.

Мутація — спонтанне перетворення (видозміна) символів (характерних особливостей) у послідовності (хромосомі). На рис. 4.1 *(в)* показано, як у результаті мутації п'ятого гена (значення 0 замінено 1) отримана нова хромосома.

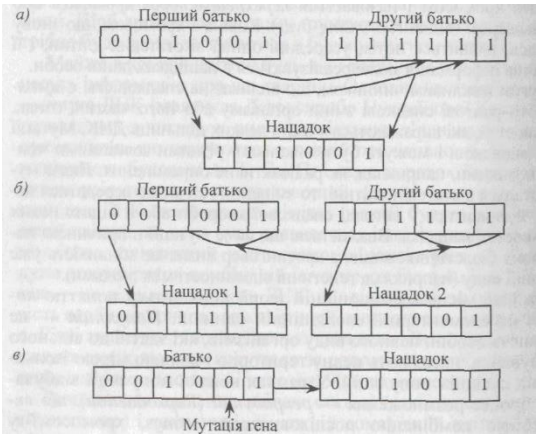


Рисунок 4.1 – Схема генеративних процесів:
а) репродукції осіб популяції; *б)* кросовера осіб популяції; *в)* мутації хромосоми

Ці процеси можуть комбінуватися для формування гібридних операторів, операцій репродукції (відтворення) і схрещування з тим, щоб бути спроможними створювати конкуренцію між популяціями.

Загальна схема генетичних алгоритмів

У концептуальному плані загальна схема генетичних алгоритмів досить проста. Спочатку генерується початкова популяція особин (індивідуумів, хромосом), тобто деякий ряд розв'язків задачі. Як правило, це робиться випадково. Потім необхідно змоделювати розмноження всередині цієї популяції. Для цього випадково підбираються кілька пар індивідуумів, проводиться схрещування хромосом у кожній парі, а отримані нові хромосоми поміщають у популяцію нового покоління. У генетичному алгоритмі зберігається засадний принцип природного добору: чим пристосованіший індивідуум (чим більше відповідне йому значення цільової функції), тим з більшою ймовірністю він буде брати участь у схрещуванні.

Потім моделюються мутації в кількох випадково вибраних особинах нового покоління, тобто змінюються деякі гени. Після цього стара популяція частково або повністю знищується і ми переходимо до розгляду наступного покоління. Популяція наступного покоління в більшості реалізацій генетичних алгоритмів містить стільки ж особин, скільки й початкова, але внаслідок відбору пристосованість (значення цільової функції) у ній в середньому вища. Операція доведення кількості особин поточної популяції до початково визначеної величини називається *редукцією.* Описані процеси відбору, схрещування і мутації повторюються вже Для цієї нової популяції.

У кожному наступному поколінні буде спостерігатися виникнення абсолютно нових розв'язків задачі. Серед них будуть як погані, так і кращі, але завдяки процедурі добору кількість кращих розв'язків буде зростати. Зауважимо, що в природі не буває абсолютних гарантій, і навіть найпристосованіший тигр може загинути від пострілу мисливця, не залишивши потомства. Імітуючи еволюцію в комп'ютері, можна уникати подібних небажаних подій і завжди зберігати життя кращому з індивідуумів поточно-^{0} покоління. Така методика називається *«стратегією елітизму»,* коли в наступне покоління відбираються особини з найкращими показниками.

Описана послідовність дій за реалізації генетичних алгоритмів може перетворюватися в різні програмні реалізації залежно від типу розв'язуваної задачі і вибраних для цього підходів. Зокрема, в низці випадків може вводитися інша, ніж описана вище, єрархія базових понять, наприклад, кожний індивідуум може характеризуватися низкою хромосом, котрі, у свою чергу, містять різні типи генів. Пояснимо на прикладі.

Нехай розглядається завдання вибору плану вкладення коштів у вибрані наперед *N* інвестиційних проектів, причому потрібно визначити обсяги вкладень коштів у кожний проект так, щоб загальний їх обсяг в усі проекти не перевищував величину *D,* а вибраний критерій ефективності, наприклад рівень рентабельності інвестицій (прибуток на капітал, ROI — Return on Investment), набував максимального значення. Розв'язуючи цю задачу за генетичним

алгоритмом, вважатимемо, що кожен індивідуум — це інвестиційний план, який містить *N* хромосом, кожна з яких являє собою вектор із нулів та одиниць — двійковий вираз обсягу вкладень у даний проект. Якщо довжина хромосоми дорівнює вісьмом двійковим розрядам, то потрібне попереднє нормування всіх чисел на інтервалі від 0 до 255 (усього значень 2^8). Такі хромосоми називаються безперервними і уможливлюють подання значень довільних числових параметрів.

Мутації безперервних хромосом випадковим способом змінюють у них один біт (ген), впливаючи у такий спосіб на значення параметра. Кросовер також можна здійснювати стандартно, об'єднуючи частини відповідних хромосом (з однаковими номерами) різних індивідуумів. Особливістю цієї задачі є те, що загальний обсяг капіталу, що інвестується, фіксований і дорівнює *D*. Очевидно, що із-за мутацій і схрещувань можна отримувати розв'язки, для реалізації яких потрібний капітал, більший або менший ніж *D*. У генетичному алгоритмі використовується спеціальний механізм аналізування таких розв'язків, що дає змогу враховувати обмеження типу «сумарний капітал = *D* " за підрахунку пристосованості індивідуума. У процесі еволюції особини з суттєвим порушенням зазначених обмежень «вимирають». Унаслідок дії алгоритму отриманий розв'язок за сумарним капіталом може не дорівнювати точно, але бути близьким до заданої величини *D*. У процесі роботи генетичного алгоритму оцінюється значення цільової функції для кожного плану і здійснюється операція редукції для всієї популяції.

Цю саму задачу можна подати і в іншій генетичній інтерпретації, якщо ввести умову, що кожний із інвестиційних проектів або цілком приймається, або відхиляється. Тоді кожний варіант плану (хромосому) можна подати у вигляді послідовності з *N* нулів та одиниць, причому, якщо на цьому місці в хромосомі стоїть одиниця, то це означає, що і-й проект *(i* - 1, 2, ..., ЛО включений у план, а якщо нуль — не включений. Популяція складається із кількох варіантів планів. Визначення допустимості планів і оцінювання їх за вибраними критеріями проводиться аналогічно.

У загальному вигляді стратегію отримання рішень за допомогою генетичних алгоритмів можна реалізувати такими кроками:

1) ініціалізуйте популяцію;

2) виберіть батьків для репродукції і оператори мутації і кросовера;

3) виконайте операції, щоб згенерувати проміжну популяцію індивідуумів і оцінити їхні придатності;

4) виберіть членів популяції для отримання нової генерації
(версії);

5) повторюйте кроки 1—3, поки не буде досягнуте деяке правило зупинки.

На рис. 4.2 показана узагальнена схема реалізації генетичного алгоритму. До його основних характеристик належать: розмір популяції, оператор кросовера і ймовірність його використання, оператор мутації і її ймовірність, оператор селекції, оператор редукції, правило (критерій) зупинки процесу виконання генетичного алгоритму. Оператори селекції, кросовера, мутації і редукції ще називають *генетичними операторами.*

Критерієм зупинки процесу здійснення генетичного алгоритму може бути одна з трьох подій:

- сформовано задану користувачем кількість поколінь;
- популяція досягла заданої користувачем якості (наприклад, значення якості всіх особин перевищило задану порогову величину);
- досягнутий деякий рівень збіжності. Тобто особини в популяції стали настільки подібними, що дальше їх поліпшення відсувається надзвичайно повільно, і тому продовження здійснення ітерацій генетичного алгоритму стає недоцільним.

Після завершення роботи генетичного алгоритму з кінцевої популяції вибирається та особина, яка дає максимальне (або мінімальне) значення цільової функції і, отже, є результатом здійснення генетичного алгоритму. За рахунок того, що кінцева популяція краща, ніж початкова, отриманий результат являє собою поліпшене рішення.

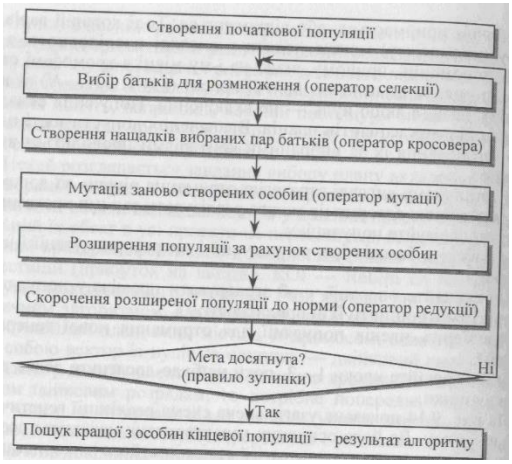


Рисунок 4.2 – Узагальнена схема реалізації генетичного алгоритму

Доступне програмне забезпечення генетичних алгоритмів

Генетичні алгоритми нині можна застосовувати в різних галузях. їх успішно використовують для розв'язування низки великих і економічно важливих задач у бізнесі і в інженерних розробках. З їх допомогою були розроблені промислові проектні рішення, що уможливили багатомільйонну економію витрат. Фінансові компанії широко використовують ці засоби у разі прогнозування розвитку фінансових ринків для управління пакетами цінних паперів. Нарівні з іншими методами генетичні алгоритми, зазвичай, використовуються для оцінювання значень безперервних параметрів моделей великих розмірностей, для розв'язування комбінаторних задач, для задач з оптимізації, що містять одночасно безперервні і дискретні параметри. Іншою галуззю їх застосування є використання в системах добування нових знань із великих баз даних, створення і навчання стохастичних мереж, навчання нейронних мереж, оцінювання параметрів у задачах багатовимірного статистичного аналізу, отримання початкових даних для виконання інших алгоритмів пошуку і оптимізації. Все це зумовило зростання заінтересованості фірм-розробників комерційного програмного забезпечення стосовно генетичних алгоритмів, що в кінцевому результаті привело до появи на ринку багатьох програмних продуктів такого виду.

Незважаючи на те, що розв'язання конкретної оптимізаційної задачі часто потребує побудови генетичного алгоритму з унікальними значеннями параметрів, низка базових властивостей цих алгоритмів залишається постійною за розв'язання абсолютно різних задач. Тому здебільшого для реалізації конкретного генетичного алгоритму не потрібно створювати окремий програмний продукт.

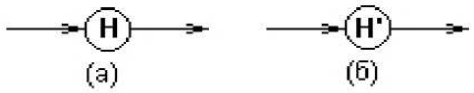
Опишемо кілька прикладів програмного забезпечення, що дає змогу реалізовувати широкий набір генетичних алгоритмів, які можна застосовувати для розв'язування найрізноманітніших задач. Змінними параметрами генетичних алгоритмів у таких додатках, зазвичай, є різні значення ймовірностей, розмір популяції і низка специфічних властивостей алгоритму. Проте реалізація генетичних операторів, як правило, єдина для всіх алгоритмів і прихована від користувача.

Пакет Evolver 4.0 компанії Palisade Corp. Пакет Evolver являє собою доповнення до програми MS Excel версій 5.0 і 7.0. При цьому Excel використовується як засіб опису початкових даних алгоритму і розрахунків у процесі його виконання. У процесі установки Evolver додає в Excel додаткову панель інструментів, яка забезпечує доступ до пакета. Якщо Evolver не запущений для виконання, то панель інструментів не відображається. У разі запуску Evolver додаток Excel запускається автоматично.

Пакет GeneHunter 1.0 компанії Ward System Group. Пакет GeneHunter багато чим схожий з пакетом Evolver. Він також є надбудовою над MS Excel версій 5.0 і 7.0 і запускається з меню «Сервіс». Цей пакет русифікований і має низку додаткових настройок для генетичних алгоритмів: включення стратегій елітизму й різноманітності. Поля вікна GeneHunter практично такі самі як і в Evolver. Однак його вікно має низку відмінностей. Для установки параметрів алгоритму служить кнопка «Параметри...». Параметри генетичного алгоритму не зберігаються автоматично з файлом Excel. Для збереження параметрів служить кнопка «Модель», після натиснення на яку з'являється відповідне діалогове вікно.

Пакет Genetic Training Option (GTO) компанії California Scientific Software. Пакет GTO є додатковою утилітою, що поставляється для нейропакета BrainMaker виробництва компанії «California Scientific Software». Він застосовується як для побудови нейронних мереж, так і для поліпшення створеної за допомогою BrainMaker мережі. Але в обох випадках окремо від BrainMaker використовуватися не може.

Генетичні алгоритми складні для створення, але прості в застосуванні — потребують від користувача тільки формалізації задачі й формування початкових даних. Така ситуація багато в чому сприяє розширенню галузей



застосування генетичних алгоритмів.

Рисунок 4.3 – Графічне зображення недавнього (а) і давнього (б) історичного стану

4.3 Порядок виконання роботи

Основні теоретичні відомості

У багатьох технічних задачах актуальною є проблема знаходження глобального оптимуму цільової функції в багатомірному просторі керованих змінних. Традиційні методи багатомірної оптимізації є методами локального пошуку та сильно залежать від вибору початкової точки пошуку. Для знаходження глобального оптимуму доцільно використовувати методи генетичного пошуку.

Генетичні методи засновані на аналогії з природними процесами селекції та генетичними перетвореннями, і поєднують комп'ютерні методи моделювання генетичних процесів у природних і штучних системах.

Традиційно до генетичних методів відносять генетичні алгоритми, генетичні стратегії, генетичне програмування та еволюційне програмування.

Генетичний пошук як метод оптимізації

Генетичний пошук містить у собі групу багатомірних, стохастичних, евристичних оптимізаційних методів, вперше запропонованих Д. Холландом у 1975 р. і заснованих на ідеї еволюції за допомогою природного відбору. Генетичні методи були отримані в процесі узагальнення й імітації в штучних системах таких властивостей живої природи, як природний відбір, пристосованість до змінюваних умов середовища, спадкування нащадками життєво важливих властивостей від батьків і т.ін.

Під стандартним генетичним методом розуміють метод для вирішення оптимізаційних задач вигляду:

$f(H) \rightarrow \min$,

де f – функція пристосованості (функція придатності, цільова функція, фітнесс-функція);

$H = \{0; 1\}^L$ – хромосома, що містить в закодованому вигляді параметри цільової функції;

L – кількість розрядів у хромосомі.

Генетичні методи в процесі пошуку використають деяке кодування множини параметрів замість самих параметрів, тому вони можуть ефективно застосовуватися для рішення задач оптимізації, визначених як на числових множинах, так і на кінцевих множинах довільної природи.

Аналогія генетичних методів з поняттями генетики

Основна концепція класичної генетики – *ген* (реально існуюча, незалежна, комбінуюча та розщеплююча при схрещуваннях одиниця спадковості) була введена І.Г. Менделем з метою пояснення спостережуваної статистики спадкування. Носіями генів у клітинному ядрі особини є нитковидні тіла – *хромосоми* (структурні елементи клітинного ядра біологічних організмів, що є носіями генів). У генетичних методах терміни “хромосома” і “особина” використаються як синоніми. Місце, що займає ген у хромосомі, називається *локусом*. Схематично можна уявити собі хромосому як прямолінійний відрізок,

а локуси – як послідовні ділянки, на які цей відрізок розбитий. Гени приймають значення, які називаються *алелями*.

Дії генів проявляються в досить великих співтовариствах організмів, що схрещуються між собою. Такі співтовариства називають *популяціями*. Популяції характеризуються набором хромосом кожного з об'єктів, сукупність яких визначає *генофонд популяції*.

Таким чином, генетичні методи запозичили з біології понятійний апарат, ідею колективного пошуку екстремуму, способи представлення генетичної інформації, способи передачі генетичної інформації в послідовності поколінь (генетичні оператори), ідею про переважне розмноження найбільш пристосованих особин.

Узагальнена схема роботи генетичних методів

Суть генетичного пошуку полягає в циклічній заміні однієї популяції наступною, більш пристосованою. Таким чином, популяція існує не тільки в просторі, але й у часі. Початкова популяція P_0 створюється на етапі ініціалізації генетичного пошуку.

Подальша робота генетичного методу представляє собою ітераційний процес виконання генетичних операторів відбору, схрещування й мутації. Генетичні оператори необхідні для того, щоб застосувати принципи спадковості й мінливості до популяції. Генетичні оператори мають властивість імовірності, тобто вони не обов'язково застосовуються до всіх рішень, що вносить додатковий елемент невизначеності в процес пошуку рішення.

Кожне рішення (хромосома) оцінюється мірою пристосованості. Пристосованість хромосоми визначається як обчислена цільова функція. Правила відбору прагнуть залишити тільки ті рішення, де досягається оптимум цільової функції. Найбільш пристосовані хромосоми одержують можливість відтворювати нащадків за допомогою схрещування з іншими хромосомами популяції. Це призводить до появи нових хромосом, які сполучають у собі деякі характеристики, наслідувані ними від батьків. Найменш пристосовані рішення з

меншою ймовірністю зможуть відтворити нащадків, у результаті чого властивості, якими вони володіли, будуть поступово зникати з популяції в процесі еволюції.

Схрещування найбільш пристосованих хромосом приводить до того, що досліджуються найбільш перспективні ділянки простору пошуку. В остаточному підсумку популяція буде сходитися до оптимального рішення задачі.

Після схрещування іноді відбуваються мутації – спонтанні зміни в генах, які випадковим чином розкидають рішення по всьому простору пошуку.

У результаті схрещування й мутації розмір популяції збільшується. Однак для наступних перетворень необхідно скоротити число хромосом поточної популяції. Як правило, наступна популяція формується з нащадків, отриманих у поточній популяції в результаті схрещування й мутації, а також елітних хромосом, що володіють найкращою пристосованістю.

Узагальнений метод генетичного пошуку можна записати в такий спосіб.

Крок 1. Встановити лічильник ітерацій (часу): $t = 0$.

Крок 2. Згенерувати початкову популяцію хромосом $P(t)$.

Крок 3. Обчислити функцію пристосованості для всіх хромосом у популяції $f(P(t))$.

Крок 4. Перевірити умови закінчення пошуку (час, число ітерацій, значення функції пристосованості і т.ін.). Якщо критерії зупину задоволені, перейти до кроку 12.

Крок 5. Збільшити лічильник ітерацій (часу): $t = t + 1$.

Крок 6. Вибрати частину популяції (батьківські хромосоми) для схрещування P'.

Крок 7. Схрестити обрані батьківські хромосоми $P'(t)$.

Крок 8. Застосувати оператор мутації до хромосом $P'(t)$.

Крок 9. Обчислити нову функцію пристосованості популяції $f(P'(t))$.

Крок 10. Вибрати хромосоми, що вижили, виходячи з рівня пристосованості.

Крок 11. Перейти на крок 4.

Крок 12. Кінець.

У наш час запропоновано багато різних генетичних методів, і в більшості випадків вони мало схожі на наведений генетичний метод. Із цієї причини під терміном “генетичні методи” мається на увазі досить широкий клас методів, часом мало схожих один на одного.

Використання генетичного пошуку для рішення практичних задач передбачає:

– вибір методу представлення вхідних даних для генетичного пошуку (кодування параметрів, що оптимізуються);

– визначення цільової функції, що використовується для оцінки хромосом;

– вибір оператору відбору хромосом, що будуть використані для генерації нових рішень за допомогою схрещування й мутації;

– вибір методів одержання нових рішень (операторів схрещування й мутації);

– завдання параметрів пошуку, таких як кількість особин у популяції, імовірнісні характеристики генетичних операторів, максимально припустима кількість ітерацій генетичного пошуку, кількість елітних особин при використанні стратегії елітизму.

Моделі генетичного пошуку

Крім узагальненої схеми функціонування генетичного методу, розглянутої вище, використовують також інші технології генетичного пошуку. Вибір моделі генетичного методу залежить від типу розв'язуваної задачі.

Виділяють наступні методи й моделі генетичного пошуку:

– канонічні моделі (*репродуктивний план Холланда,* генетичний метод Девиса, генетичний метод Гольдберга);

– модель *Genitor (Д. Уітлі);*

– гібридні *генетичні* методи;

– модель CHC;

– генетичний метод зі змінним часом життя особин;

– мобільний генетичний метод;

– паралельні й багаторівневі генетичні методи (однопопуляційні генетичні методи, острівна модель, дрібноструктурні генетичні методи, ієрархічні гібриди);

– генетичний пошук зі зменшенням розміру популяції.

Ініціалізація та запуск генетичного пошуку

Кодування параметрів, що оптимізуються

Будь-який організм може бути представлений своїм *фенотипом*, що фактично визначає, чим є об'єкт у реальному світі, і *генотипом*, що містить всю інформацію про об'єкт на рівні хромосомного набору. При цьому кожний *ген*, тобто елемент інформації генотипу, має своє відображення у фенотипі. Таким чином, для розв’язку задач необхідно представити кожну ознаку об'єкта у формі, що підходить для використання в генетичному методі. Все подальше функціонування механізмів генетичного методу відбувається на рівні генотипу, що дозволяє обходитися без інформації про внутрішню структуру об'єкту, що й обумовлює широке застосування генетичного пошуку в самих різних задачах.

За методами представлення генів хромосоми можна умовно розділити на три групи:

1. *Бінарні хромосоми* – хромосоми, гени яких можуть приймати значення 0, або 1.

2. *Числові хромосоми* – гени можуть приймати значення в заданому інтервалі.

Числові хромосоми можна розділити на гомологічні та негомологічні.

Гомологічними називають хромосоми, що мають загальне походження, морфологічно та генетично подібні, і тому не утворюють неприпустимих рішень при застосуванні стандартних генетичних операторів. У гомологічних числових хромосомах кожний ген може приймати цілі значення в заданому інтервалі. Для різних генів можуть бути задані різні інтервали. Бінарна хромосома є

гомологічною числовою хромосомою, кожний ген якої може приймати цілі значення в інтервалі [0, 1].

У *негомологічних* хромосомах гени можуть приймати значення в заданому інтервалі; при цьому інтервал однаковий для всіх генів, але в хромосомі не може бути двох генів з однаковим значенням. Для негомологічних хромосом застосовуються різні спеціальні генетичні оператори, що не створюють неприпустимих рішень. Негомологічні хромосоми, як правило, застосовуються при розв'язку задач комбінаторної оптимізації.

Векторні хромосоми – хромосоми, гени яких представляють собою вектор цілих чисел.

Ген у векторних хромосомах має властивості негомологічної хромосоми, тобто числа у векторі можуть приймати значення в заданому інтервалі, і вектор не може містити двох однакових чисел. Проте, хоча гени у векторних хромосомах негомологічні, самі векторні хромосоми є гомологічними.

Процес кодування параметрів, що оптимізуються, можна виконати в наступній послідовності кроків.

Крок 1. Визначення параметрів, що оптимізуються, – генів.

Крок 2. Вибір числа розрядів у кожному гені.

Крок 3. Вибір методу кодування.

Варто врахувати, що занадто велика довжина кодування прискорює процес збіжності всіх членів популяції до кращого знайденого рішення. Часто такий ефект є небажаним, оскільки при цьому більша частина простору пошуку залишається недослідженою. Передчасна збіжність може не привести до оптимального рішення, крім того, швидка збіжність до однієї області не гарантує виявлення декількох рівних екстремумів. До того ж застосування довгих кодувань зовсім не гарантує, що знайдене рішення буде мати необхідну точність, оскільки цього, в принципі, не гарантує сам генетичний метод.

Тому в питанні вибору оптимальної довжини кодування потрібно досягти деякого компромісного рішення – з одного боку довжина хромосоми повинна бути досить великою, щоб все-таки забезпечити швидкий пошук, з іншого боку

– по можливості малою, щоб не допускати передчасної збіжності й залишити методу шанс відшукати кілька оптимальних значень.

Наведемо варіанти кодування генів у деяких задачах, розв'язуваних за допомогою генетичних методів:

– оптимізація функцій: гени – незалежні змінні;

– апроксимація: гени – параметри-константи апроксимуючих функцій;

– задача відбору інформативних ознак: гени ідентифікують значимість відповідних ним ознак (наприклад, якщо значення гену дорівнює одиниці, то відповідна йому ознака вважається інформативною);

– настроювання ваг штучної нейронної мережі: гени відповідають синоптичним вагам нейронів;

– штучне життя (Artificial Life): гени відповідають характеристикам особини (сила, швидкість, і т.ін.), також повинні бути незмінні гени, що позначають тип особини (рослина або тварина);

– задача про найкоротший шлях: гени – пункти пересування. Вся хромосома представляє собою маршрут з початкової точки в кінцеву, причому не завжди існуючий.

Невдалий вибір упорядкування та кодування бітів у хромосомі може викликати передчасну збіжність до локального оптимуму. Для подолання цього недоліку можна вибирати спосіб кодування, ґрунтуючись на додаткової інформації про задачу.

Варто відзначити, що використання різних варіантів кодування розподіляє точки в просторі пошуку по-різному. У найбільш розповсюдженій різновиді генетичного пошуку для представлення генотипу об'єкту використовуються *бітові рядки*. При цьому кожному атрибуту об'єкту у фенотипі відповідає один ген у генотипі об'єкта. Ген є бітовим рядком, найчастіше фіксованої довжини, що являє собою значення цієї ознаки.

Кількість розрядів r у гені для кодування ознаки визначається за формулою:

$$r = \text{ceil}\left(\log_2\left(\frac{w_{max} - w_{min}}{\varepsilon}\right)\right),$$

де ceil(x) – найближче більше або рівне x ціле число;

w_{max} і w_{min} – максимально й мінімально можливі значення ознаки (параметра, незалежної змінної);

ε – задана похибка визначення оптимального значення ознаки.

Розрядність хромосоми L визначається як сума розрядностей генів. У випадку, якщо задані однакові значення w_{max}, w_{min} і ε для всіх n генів, розрядність хромосоми може бути обчислена за формулою:

$L = n \cdot r$.

Після того, як обрані параметри, їх кількість та розрядність, необхідно вирішити, як безпосередньо записувати дані, тобто вибрати метод кодування. Можна використати *звичайне кодування* або *коди Грея*. Незважаючи на те, що використання кодів Грея призводить до кодування/декодування даних, вони дозволяють уникнути деяких проблем, які з'являються в результаті звичайного кодування.

Перевага коду Грея в тому, що якщо два числа відрізняються на 1, то і їхні двійкові коди відрізняються тільки на один розряд, а у двійкових кодах не все так просто. Так, наприклад, числа 7 і 8 у бітовому поданні відрізняються в чотирьох позиціях ($7_{10}=0111_2$, $8_{10}=1000_2$), що затрудняє функціонування генетичного методу й збільшує час, необхідний для його збіжності, а в коді Грея ці числа відрізняються всього на одну позицію ($7_{10}=0100_Г$, $8_{10}=1100_Г$).

Варто відзначити, що кодувати й декодувати у коди Грея досить зручно.

Кодування/декодування з бінарного коду в код Грея можна виконати в такий спосіб.

Крок 1. Скопіювати старший розряд кодуємого (декодуємого) числа в старший розряд декодуємого (кодуємого) числа.

Крок 2. Виконати перетворення за формулами:

– із двійкового коду в код Грея: $G[i] = \text{XOR}(B[i+1], B[i])$;

– з коду Грея у двійковий: $B[i] = \mathrm{XOR}(B[i+1], G[i])$,

де $G[i]$ – i-ий розряд коду Грея;

$B[i]$– i-ий розряд бінарного коду.

Наприклад, послідовність чисел від 0 до 15 у двійковому коді: {0000, 0001 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001 1010, 1011, 1100, 1101, 1110, 1111}, а в кодах Грея: {0000, 0001 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101 1111, 1110, 1010, 1011, 1001, 1000}.

Кодування ознак, яким відповідають числа із плаваючою комою.

Найпростіший спосіб кодування – використання бітового представлення.

Однак такий варіант має ті ж недоліки, що й при кодуванні цілих чисел. Тому на практиці застосовується наступна послідовність дій.

Крок 1. Розбивається весь інтервал допустимих значень ознаки на ділянки з необхідною точністю.

Крок 2. Приймається значення гена як ціле число, що визначає номер інтервалу (використовуючи код Грея).

Крок 3. В якості значення параметра приймається число, що є серединою цього інтервалу.

Завдання цільової функції

Цільова функція – це функція, оптимум якої необхідно знайти. Генетичний метод вимагає, щоб хромосоми оцінювалися за допомогою цільової функції (фітнесс-функцій, функції пристосованості, функції оцінки) задачі.

Наприклад, для задачі апроксимації, цільовою функцією може бути середнє відхилення значень вихідного параметру, розрахованих за утвореною моделі, від його реальних значень.

Можна відзначити, що обчислення фітнесс-функції – один з найбільш важливих етапів генетичного пошуку.

Тому при виборі цільової функції потрібно враховувати наступне.

1. Функція пристосованості повинна бути адекватна задачі. Це означає, що для успішного пошуку необхідно, щоб розподіл значень фітнесс-функціх збігався з розподілом реальної якості рішень (не завжди “якість” рішення еквівалентна його оцінці за фітнесс-функцією).

2. Фітнесс-функція повинна мати рельєф. Крім того, рельєф повинен бути різноманітним. Це означає, що генетичний метод має мало шансів на успіх, якщо на поверхні фітнесс-функції є величезні “плоскі” ділянки, тому що це приводить до того, що більшість рішень (хромосом) у популяції при різних генотипах не будуть відрізнятися фенотипом. Тобто, незважаючи на те, що рішення розрізняються, вони мають однакову оцінку, а значить метод не має можливості вибрати краще рішення та вибрати напрямок подальшого розвитку.

3. Фітнесс-функція повинна вимагати мінімум ресурсів, тому що її обчислення є найбільш часто виконуваним етапом методу, і тому складність обчислення фітнесс-функції має істотний вплив на швидкість роботи методу.

4. У випадку, якщо цільова функція містить ділянки, що представляють собою так зване “вузьке горло” (різкий стрибок або спад), необхідно врахувати, що генетичний пошук може не знайти глобального екстремуму, що розташований у вузькому горлі. Для підвищення якості генетичного пошуку при такій цільовій функції можна рівномірно формувати початкову популяцію на всьому інтервалі припустимих значень змінних.

Ініціалізація

Стандартні генетичні методи починають свою роботу з ініціалізації, тобто формування *початкової популяції* P_0 – кінцевого набору допустимих рішень задачі:

$$P_0 = \{H_1, H_2, \ldots, H_N\},$$

де N – розмір популяції;

$H_j = \{h_{1j}, h_{2j}, \ldots, h_{Lj}\}$ – хромосома, що складається з L генів;

$\min_i \le h_{ij} \le \max_i$, $\min_i$ і $\max_i$ – мінімальне й максимальне значення i-го параметра в розв'язуваній за допомогою генетичного методу задачі.

Ці рішення можуть бути обрані випадковим образом або введені користувачем. Вибір початкової популяції не має значення для збіжності процесу в асимптотиці, однак, формування гарної початкової популяції (наприклад, із множини локальних оптимумів) може помітно скоротити час досягнення глобального оптимуму. Таким чином, при наявності необхідної інформації завдання початкової популяції користувачем є кращим.

Найчастіше розмір початкової популяції вибирається в інтервалі 20–100 особин.

Стратегія створення початкової популяції може бути різною. Відомі наступні стратегії.

1. *Стратегія "ковдри"* – вихідна множина містить всі можливі варіанти рішень. Стратегія "ковдри" має наступні недоліки:

– у багатьох випадках неможливо здійснити повний перебір;

– з погляду адаптивного розвитку не представляє цінності, тому що часто вже в першому поколінні рішень відшукується оптимальне рішення.

2. *Стратегія "фокусування"* – стартова множина рішень включає різновиди одного рішення.

Стратегія "фокусування" застосовується в тих випадках, коли є припущення, що деяке рішення є різновидом відомого субоптимального. Тоді шляхом поступових незначних змін існуючого рішення можна одержати більш якісне субоптимальне рішення.

Для більшості задач оптимізації неприйнятні стратегії "ковдри" (внаслідок проблематичності повного перебору) і фокусування (відсутня чітка залежність якості рішення від параметрів рішення).

3. *Стратегія "дробовика"* – генерується досить велика множина рішень.

Дана стратегія може бути реалізована одним з трьох способів.

3.1. Рівномірне формування початкової популяції.

3.2. Випадкове формування початкової популяції.

3.3. Комплементарне формування початкової популяції, яке виконується за два кроки.

Крок 1. Сформувати випадковим чином першу половину початкової популяції.

Крок 2. Сформувати другу половину початкової популяції шляхом додавання хромосом, протилежних (одиниці заміняються нулями) хромосомам в першій половині популяції.

Відбір

Оператор відбору (вибору, селекції) представляє собою оператор, що на основі значення цільової функції вибирає хромосоми таким чином, щоб з ненульовою ймовірністю будь-який елемент популяції міг би бути обраний у якості одного з батьків при схрещуванні.

Найпоширенішими є наступні оператори відбору:

– пропорційний відбір (пропорційно-імовірнісний відбір);

– відбір ранжируванням;

– турнірний відбір;

– відбір з використанням порогу.

Пропорційний відбір

Даний вид відбору складається з наступної послідовності кроків.

Крок 1. Обчислити пристосованість кожної особини f_j.

Крок 2. Знайти середню пристосованість у популяції $f_{\text{ср}}$ як середнє арифметичне значень пристосованості всіх особин:

$$f_{cp} = \frac{1}{N} \sum_{j=1}^{N} f_j \ .$$

Крок 3. Для кожної особини обчислити відношення $P_s(j) = \frac{f_j}{f_{cp}}$.

Крок 4. Залежно від величини $P_s(j)$ сформувати масив особин, допущених до схрещування.

Формування масиву допущених до схрещування особин (крок 4) можна здійснити двома шляхами:

Перший шлях (стохастичний залишковий відбір): якщо $P_s(j) > 1$, то особина вважається добре пристосованою та допускається до схрещування.

Наприклад, якщо дріб $P_s(j) = 2{,}36$, то дана особина має подвійний шанс на схрещування й буде мати ймовірність рівну 0,36 третього схрещування. Якщо ж пристосованість дорівнює 0,54, то особина візьме участь у єдиному схрещуванні з імовірністю 0,54.

Другий шлях: після знаходження відносини $P_s(j)$ відбувається відбір (із заміщенням) всіх N особин для подальшої генетичної обробки, відповідно до величини $P_s(j)$.

Найпростіший пропорційний відбір – *рулетка* – відбирає особини за допомогою N запусків рулетки.

Колесо рулетки містить по одному сектору для кожного члена популяції. Розмір j-го сектору пропорційний відповідній величині $P_s(j)$. Особина одержує можливість створення нащадків, якщо випадково згенероване число в межах від 0 до 2π попадає в сектор, що відповідає цій особини. При такому відборі члени популяції з більш високою пристосованістю з більшою ймовірністю будуть частіше вибиратися, чим особини з низькою пристосованістю.

При реалізації відбору рулеткою доцільно замінити колесо рулетки інтервалом [0;1] у зв'язку з тим, що в такому випадку немає необхідності обчислювати ширину кожного сектора – у цьому випадку кожній особині ставиться у відповідність напівінтервал $[x_{j-1}; x_j)$, де $x_{j-1} - x_j = P_s(j)$, а $x_0 = 0$ (при цьому $x_N = 1$). У наступне покоління переходить особина з номером j, де j: $x_{\text{r nd}} \in [x_{j-1}; x_j)$, а число $x_{\text{r nd}}$ повертається щораз випадковою функцією з рівномірним розподілом щільності ймовірності на відрізку [0;1].

Схема рулетки може давати дуже великі помилки, у тому розумінні, що кінцеве число нащадків даної особини може сильно відрізнятися від очікуваного. Кінцеве число наближається до очікуваного тільки в популяціях дуже більших розмірів.

Відбір ранжируванням

Відбір ранжируванням виконується за 4 кроки.

Крок 1. Обчислити пристосованість кожної особини f_j.

Крок 2. Відсортувати (ранжирувати) популяцію по зростанню пристосованості особин.

Крок 3. Для кожної особини обчислити величину $P_s(j)$. Для цього використати один із двох видів ранжирування.

а) лінійне ранжирування: $P_s(j) = \frac{1}{N}\left(\eta_{\max} - (\eta_{\max} - \eta_{\min})\frac{j-1}{N-1}\right)$, де $\eta_{\max} \in [1; 2]$, $\eta_{\min} = 2 - \eta_{\max}$.

б) рівномірне ранжирування: $P_s(j) = \begin{cases} \frac{1}{\mu}, 1 \le i \le \mu, \\ 0, \ \mu < i \le N, \end{cases}$

де μ – деяке фіксоване число перших членів популяції.

Крок 4. Залежно від величини $P_s(j)$ відібрати певну частину особин для схрещування.

Турнірний відбір

Турнірний відбір реалізує k турнірів, щоб вибрати k особин. Кожний турнір складається із двох етапів.

Етап 1. Вибір m елементів з популяції.

Етап 2. Вибір кращої особини серед особин, відібраних на попередньому етапі.

Розмір групи особин, що відбираються для турніру, часто дорівнює 2. У цьому випадку говорять про *парний турнір*. Взагалі ж m називається *чисельністю турніру*.

Турнірний відбір має певні переваги перед пропорційним, тому що не втрачає своєї вибірковості, коли в ході еволюції всі елементи популяції стають приблизно рівними за значенням цільової функції.

Відбір з використанням порогу

Відбір з використанням порогу (відбір усіканням) виконується в наступній послідовності.

Крок 1. Обчислити пристосованість кожної особини f_j.

Крок 2. Відсортувати популяцію по зростанню пристосованості особин.

Крок 3. Задати поріг $П \in [0;1]$. Поріг визначає, яка частка особин, починаючи з найпершої (самої пристосованої), буде брати участь у відборі. В принципі, поріг можна задати й числом, більшим за одиницю, тоді він буде просто дорівнює числу особин з поточної популяції, допущених до відбору.

Крок 4. Серед особин, що потрапили під значення порога, випадковим образом N раз вибирати саму везучу й записувати її в проміжний масив, з якого потім вибираються особини безпосередньо для схрещування.

Через те, що в цій стратегії використається відсортована популяція, час її роботи може бути більшим для популяцій великого розміру й залежати також від алгоритму сортування.

Схрещування

У теорії еволюції важливу роль відіграє те, яким чином ознаки батьків передаються нащадкам. У генетичних методах за передачу ознак батьків нащадкам відповідає оператор схрещування (кроссинговер, кроссовер, рекомбінація). Цей оператор моделює процес схрещування особин і визначає передачу ознак батьків нащадкам.

Метою оператору схрещування є породження з наявної множини рішень нової, у якій кожна хромосома буде нащадком деяких двох елементів попередньої популяції, тобто нести в собі частково інформацію кожного батька. Допускається ситуація, коли обидва батьки представлені одним елементом популяції.

Вибір батьківської пари

Вибираючи щораз для схрещування найбільш пристосовані особини, можна з певним ступенем впевненості стверджувати, що нащадки будуть або не набагато гіршими, ніж батьки, або кращими за них.

Існує кілька способів вибору батьківської пари.

Випадковий вибір батьківської пари (*панміксія*) – це найпростіший підхід, коли обидві особини, які утворять батьківську пару, випадковим чином вибираються із всієї популяції, причому будь-яка особина може стати членом декількох пар. Незважаючи на простоту, такий підхід універсальний для розв’язку різних класів задач. Однак він досить критичний до чисельності популяції, оскільки ефективність методу, що реалізує такий підхід, знижується з ростом чисельності популяції.

Крок 1. Для вибору пари батьків задається ймовірність схрещування P_c.

Крок 2. Довільним чином нумеруються всі представники вихідної популяції.

Крок 3. Вибір першого батька: починаючи з першого рішення, проглядається популяція доти, поки випадково обране число з інтервалу [0, 1] не буде меншим, ніж P_c. Елемент, для якого виконується така умова, стає першим батьком.

Крок 4. Відбувається перегляд популяції, починаючи з наступному після першого батька рішення, поки знову випадково обране число не буде меншим, ніж P_c. Елемент, для якого виконується така умова, стає другим батьком.

Описаним способом складаються пари доти, поки не вибереться потрібна кількість пар батьків.

Конкретне значення P_c залежить від розв'язуваної задачі, і в загальному випадку лежить в інтервалі [0,6; 0,99].

Незважаючи на простоту, такий підхід універсальний для розв’язування різних класів задач. Однак він досить критичний до чисельності популяції, оскільки ефективність методу, що реалізує такий підхід, знижується з ростом чисельності популяції.

Інший метод випадкового вибору батьківської пари може бути представлений у вигляді наступної послідовності кроків.

Крок 1. Розбити популяцію випадковим чином на два масиви (підпопуляції) одного розміру.

Крок 2. Відсортувати кожну підпопуляцію.

Крок 3. Сформувати пари для схрещування з особин, що мають однаковий ранг (номер) у підпопуляціях.

Крок 4. Допустити до схрещування пари, для яких випадково згенероване в інтервалі [0;1] число буде перевищувати задану ймовірність схрещування.

Селективний спосіб вибору особин у батьківську пару полягає в тому, що батьками можуть стати тільки ті особини, значення пристосованості яких не менше середнього значення пристосованості по популяції, при рівній імовірності таких кандидатів утворити батьківську пару.

Такий підхід забезпечує більш швидку збіжність генетичного пошуку. Однак через швидку збіжність селективний вибір батьківської пари не підходить тоді, коли ставиться задача визначення декількох екстремумів, оскільки для таких задач метод, як правило, швидко збігається до одного з рішень.

Крім того, для деякого класу задач зі складним ландшафтом фітнесс-функції швидка збіжність може перетворитися в передчасну збіжність до квазіоптимального розв'язку. Цей недолік може бути частково компенсований використанням відповідного механізму відбору, який би "гальмував" занадто швидку збіжність методу.

Інші два способи формування батьківської пари – це *інбридинг* та *аутбридинг*. Обоє ці методи побудовані на формуванні пари на основі близького й далекого "споріднення", відповідно. Під "спорідненням" тут розуміється відстань між членами популяції як у сенсі евклідової (геометричної) відстані особин у просторі параметрів (для фенотипів), так і у сенсі відстані Хеммінгу між хромосомними наборами особин (для генотипів).

Евклідова відстань $R^{(jk)}$ між j-ою та k-ою особинами популяції визначається за формулою:

$$R^{(jk)} = \sqrt{\sum_{i=1}^{p} \left(x_i^{(j)} - x_i^{(k)} \right)^2} ,$$

де p – кількість параметрів (генів) особини;

$x_i^{(j)}$ – i-ий параметр у незакодованому вигляді j-ої особини.

Відстань Хеммінгу $H^{(jk)}$ між j-ою та k-ою особинами популяції визначається як кількість різних бітів в однакових позиціях j-ої та k-ої хромосом.

Інбридинг складається з двох етапів:

1. Перший член пари вибирається випадково.

2. Другим батьком з більшою ймовірністю буде максимально близька до першого особина.

Один з варіантів процедури інбридингу може бути реалізований у такий спосіб.

Крок 1. Вибрати випадковим чином першого батька.

Крок 2. Вибрати з поточної популяції випадковим чином групу з C хромосом (C = 1%–15% від розміру популяції).

Крок 3. Розрахувати *Евклідову* відстань від хромосоми, отриманої на першому кроці, до кожної із C відібраних на другому кроці хромосом.

Крок 4. В якості другого батька вибрати найближчу до першого батька хромосому.

Аутбридинг формує батьківські пари з максимально далеких особин.

Використання генетичних інбридингу й аутбридингу є ефективним для багато екстремальних задач. Однак два цих способи по-різному впливають на поводження генетичного методу. Інбридинг можна охарактеризувати властивістю концентрації пошуку в локальних вузлах, що фактично призводить до розбиття популяції на окремі локальні групи навколо підозрілих на екстремум ділянок ландшафту. Аутбридинг, навпаки, спрямований на попередження збіжності методу до вже знайдених рішень, змушуючи метод переглядати нові, недосліджені області.

Оператори схрещування

При ***n-точковому*** схрещуванні:

1. Випадково обираються n точок розриву, що приводить до розбиття вихідних векторів на $n + 1$ частин різної довжини.

2. Обмінюються у вихідних хромосомах ділянки з парними номерами, а ділянки з непарними залишаються без змін:

n-точкове схрещування може застосовуватися для бінарних, векторних і гомологічних числових хромосом.

Класичним варіантом такого схрещування є одноточечне схрещування, при якому:

1. Випадковим чином визначається точка в середині хромосоми. Ця точка називається точкою розриву (точкою *схрещування,* crossover point).

2. В обраній точці обидві хромосоми діляться на дві частини й обмінюються ними. У результаті утворюються два нащадки.

Даний тип схрещування називається одноточечним, тому що при ньому батьківські хромосоми розрізаються тільки в одній випадковій точці.

При двохточечному схрещуванні в хромосомі випадково вибираються вже дві точки схрещування. Ліву точку будемо вважати першою, а праву – другою. Перший нащадок формується із частин першого батька, розташованих лівіше від першої точки схрещування й правіше від другої точки, і частини другого батька, розташованої між першою й другою точками схрещування. Другий нащадок формується із лівої та правої частин другого батька й центральної частини першого батька.

Обчислювальні експерименти показали, що навіть для простих функцій не можна говорити про перевагу того або іншого оператора. Більше того, використання механізму випадкового вибору одно- або двохточечного схрещування для кожної конкретної батьківської пари часом виявляється більше ефективним, ніж детермінований підхід до його вибору, оскільки досить важко апріорно визначити, який із двох операторів більше підходить для кожного конкретного виду функції пристосованості.

Однорідне схрещування (uniform crossover) генерує нащадка шляхом випадкової передачі йому генетичної інформації від батьків. Генерація нащадка виконується в такий спосіб.

Крок 1. Встановити лічильник бітів (генів) нащадка: $j = 1$.

Крок 2. Визначити хромосому з батьківської пари, що передасть значення свого j-го гена нащадкові: $n = \text{round}(\text{rand}[0;1])$, де rand[0;1] – випадково згенероване число в інтервалі [0;1]; round(A) – округлене значення числа A.

Крок 3. Виконати: $h_{j\text{п}} = h_{jn}$, де $h_{j\text{п}}$ – значення j-го гена нащадка, h_{jn} – значення j-го гена n-го батька.

Крок 4. Виконати: $j = j + 1$.

Крок 5. Якщо $j > L$, де L – довжина хромосоми, тоді виконати перехід на крок 6. У противному випадку перейти на крок 2.

Крок 6. Кінець.

Рівномірне схрещування може застосовуватися для бінарних, гомологічних числових і векторних хромосом. Таке схрещування зручно застосовувати в тому випадку, коли вже отримані індивіди з гарними спадкоємними ознаками, і їх необхідно закріпити в поточній популяції. Рівномірне схрещування виконується в наступній послідовності кроків.

Крок 1. Випадковим чином задається маска схрещування, що представляє собою рядок з нулів та одиниць, довжини, що дорівнює довжині хромосом.

Крок 2. Формування першого нащадка.

Одиниця на конкретній позиції в масці схрещування означає, що елемент, що знаходиться на тому ж місці в першого батька, необхідно помістити на це місце в першій дитині. Нуль на цій позиції в масці схрещування означає, що елемент, що знаходиться на тому ж місці в другого батька, необхідно помістити на це місце першій дитині.

Крок 3. Формування другого нащадка.

Якщо першого батька вважати другим, а другого – першим, то аналогічно кроку 2 можна одержати другого нащадка.

Варто відзначити, що маска схрещування може бути одна для всіх хромосом, або своя для кожної пари батьків.

При ***порівняльному схрещуванні*** в хромосомах батьків порівнюються всі біти. Якщо на однакових позиціях обох батьків знаходяться однакові біти (0 і 0 або 1 і 1), то нащадкам присвоюються ті ж значення відповідних бітів. Якщо на однакових позиціях батьків розташовані гени із різними значеннями, тоді значення відповідних генів нащадків визначають за допомогою генератору випадкових чисел.

Арифметичне (диференціальне) схрещування є найбільш вдалим для пошуку оптимуму функції багатьох дійсних змінних.

Нехай H_1 і H_2 – це два індивідууми в популяції, тобто дійсні вектори, від яких залежить цільова функція. Тоді нащадок $H_п$ обчислюється за формулою $H_п = H_1 + k \cdot (H_1 – H_2)$, де k – це деякий дійсний коефіцієнт (який може залежати від $||H_1 – H_2||$ – відстані між векторами). У цій моделі мутація – це додавання до індивідуума випадкового вектора малої довжини. Якщо вихідна функція неперервна, то ця модель працює добре, а якщо вона ще й гладка, те – відмінно.

При ***діагональному схрещуванні*** для схрещування R батьків випадковим чином вибирається $(R – 1)$ однакових точок схрещування в кожному з них. R нащадків отримують шляхом комбінування відповідних елементів батьків по діагоналі.

Крок 1. Вибрати випадковим чином R батьківських хромосом для схрещування.

Крок 2. Вибрати випадковим чином $(R – 1)$ точок схрещування в хромосомах, розділивши тим самим кожну з них на R сегментів.

Крок 3. Встановити: $r = 1$.

Крок 4. Сформувати r-го нащадка.

Крок 4.1 Встановити: $k = 1$.

Крок 4.2. Сформувати k-ий сегмент r-го нащадка, взявши g-ий сегмент із k-ої хромосоми-батька, де

$$g = \begin{cases} k + r - 1, & \text{якщо } k + r - 1 \le R, \\ k + r - 1 - R, & \text{якщо } k + r - 1 > R. \end{cases}$$

Крок 4.3. Виконати: $k = k + 1$.

Крок 4.4. Якщо $k \le R$, виконати перехід до кроку 4.2.

Крок 5. Виконати: $r = r + 1$.

Крок 6. Якщо $r \le R$, виконати перехід до кроку 4.

Крок 7. Кінець.

Слід зазначити, що різні типи схрещування мають загальну позитивну властивість: вони контролюють баланс між подальшим використанням уже знайдених гарних підобластей простору пошуку та дослідженням нових підобластей. Це досягається за рахунок неруйнування загальних блоків усередині хромосом-батьків і одночасного дослідження нових областей в результаті обміну частинами хромосом.

Спільне використання операторів відбору та схрещування призводить до того, що області простору з кращою в середньому оптимальністю, вміщують більше членів популяції, чим інші. Тому, оператор схрещування є найбільш критичним із всіх операторів генетичних методів з погляду одержання глобальних результатів.

У малих популяціях краще застосовувати більш руйнівні варіанти схрещування (багатоточечне та однорідне), а в великих популяціях краще працює двохточечне.

Мутація

Оператор мутації полягає в зміні генів у випадково обраних позиціях. На відміну від операторів відбору та схрещування, які використовуються для поліпшення структури хромосом, метою оператора мутації є диверсифікація, тобто підвищення розмаїтості пошуку й введення нових хромосом у популяцію для того, щоб більш повно досліджувати простір пошуку. Оскільки число членів популяції P звичайно вибирається значно меншим у порівнянні із загальним числом (2^L) можливих хромосом у просторі пошуку, то в силу цього вдається

досліджувати лише його частину. Отже, мутація ініціює розмаїтість у популяції, дозволяючи переглядати більше рішень у просторі пошуку й виходити в такий спосіб з локальні екстремумів в процесі пошуку.

В результаті виконання мутації з ненульовою ймовірністю чергове рішення може перейти в будь-яке інше рішення.

Вибір хромосом для мутації відбувається в такий спосіб.

Крок 1. Нумеруються довільним чином всі хромосоми H_j вихідної популяції.

Крок 2. Починаючи з першої хромосоми, проглядається вся популяція, при цьому кожній хромосомі H_j ставляться у відповідність випадкові числа x_j з інтервалу [0;1).

Крок 3. Якщо число x_j виявляється меншим за ймовірність мутації $P_м$, то поточна хромосома H_j піддається мутації.

Серед рекомендацій з вибору ймовірності мутації нерідко можна зустріти варіанти $1/L$ або $1/N$, де L – довжина хромосоми, N – розмір популяції. Імовірність мутації значно менше ймовірності схрещування й рідко перевищує 1%.

Необхідно відзначити, що оператор мутації є основним пошуковим оператором і відомі такі методи, що не використають інших операторів крім мутації.

Проста мутація

Проста мутація використовується для бінарних, гомологічних числових і векторних хромосом. Суть її полягає у внутригенній мутації. При цьому в хромосомі випадковим чином вибирається ген, а потім відбувається його випадкова зміна.

Алгоритм простої мутації для бінарних і гомологічних числових хромосом може складатися з наступних кроків.

Крок 1. Скопіювати батьківську хромосому в хромосому- нащадка.

Крок 2. Вибрати випадковим чином ген для мутації.

Крок 3. У заданому інтервалі припустимих значень гена вибрати нове значення гена, що не дорівнює поточному.

Для ***векторних хромосом*** проста мутація відбувається шляхом внесення змін у порядок елементів усередині обраного гена.

Крок 1. Скопіювати батьківську хромосому в хромосому- нащадка.

Крок 2. Вибрати випадковим чином ген для мутації.

Крок 3. Вибрати випадковим чином точку мутації усередині мутуючого гена.

Крок 4. Зробити обмін значеннями між розрядами мутуючого гена, що перебувають безпосередньо ліворуч і праворуч від точки мутації.

Відомий також модифікований оператор простої мутації, що називається ***точковою мутацією***, який відрізняється тим, що в хромосомі мутує не один, а кілька генів із заданою ймовірністю. Такий оператор використовується для бінарних, гомологічних числових і векторних хромосом. Алгоритм даного оператору наведений нижче.

Крок 1. Скопіювати батьківську хромосому в хромосому- нащадка.

Крок 2. Встановити $i = 1$.

Крок 3. Випадковим чином згенерувати число хі з інтервалу [0;1).

Крок 4. Якщо число x_i виявляється меншим ймовірності мутації гену $P_{мг}$, то виконати мутацію гена h_i.

Крок 5. Встановити $i = i + 1$.

Крок 6. Якщо $i < (L + 1)$, де L – довжина хромосоми, то виконати перехід на крок 3.

Крок 7. Кінець.

Крім того, відома множина різних варіантів операторів мутації. Більшість із них використовують комбінації наступних ідей.

1. Так само як і схрещування, мутація може проводитися не тільки по одній випадковій точці. Можна вибирати деяку кількість точок у хромосомі для інверсії, причому їхнє число також може бути випадковим.

2. Піддається мутації відразу деяка група послідовних генів.

3. Спільне застосування випадкової мутації та часткової перебудови рішення алгоритмами локального пошуку. Застосування локального спуску дозволяє генетичному методу зосередитися тільки на локальних оптимумах. Множина локальних оптимумів може виявитися експоненціально більшим і на перший погляд здається, що такий варіант методу не буде мати великих переваг. Однак експериментальні дослідження розподілу локальних оптимумів свідчать про високу концентрацію їх у безпосередній близькості від глобального оптимуму. Це спостереження відоме як гіпотеза про існування “великої долини” для задач на мінімум або “центрального гірського масиву” для задач на максимум.

Мутація гомологічних числових хромосом

Такі види мутацій полягають в зміні обраного для мутації гена h_{ji} (або всієї хромосоми H_j) на деяку величину Δh_{ij}, розраховану за певними методами:

$h_{ij}' = h_{ij} + \Delta h_{ij}$,

де h_{ij} – ген до мутації; h_{ij}' – ген після мутації

1. ***Нерівномірна (non-uniform) мутація*** до обраного для мутації i-го гену h_{ji} хромосоми H_j застосовується за формулою:

$$h_{ji}' = \begin{cases} h_{ji} + \Delta\left(t, \max_i - h_{ji}\right) \text{ з ймовірністю } 0{,}5, \\ h_{ji} - \Delta\left(t, h_{ji} - \min_i\right) \text{ з ймовірністю } 0{,}5, \end{cases}$$

де $\Delta(t, y) = y \cdot \left(1 - r^{(1-t/T)^k}\right)$;

r = rand[0; 1] – випадково згенероване число в інтервалі [0; 1];

t – номер поточної ітерації;

T – максимальна кількість ітерацій;

k – параметр, що визначає ступінь однорідності (рівномірності);

$\min_i$ і $\max_i$ – мінімальне й максимальне значення i-го параметру в розв'язуваній за допомогою генетичного методу задачі.

Крім того, нерівномірна мутація i-го гену j-ої хромосоми h_{ji} може бути виконана за формулою:

$$h_{ji}' = \begin{cases} h_{ji} + \Delta(t, \max_i - h_{ji}), & \text{якщо } f(H_j + \Delta(t, \max_i - h_{ji})) \geq f(H_j - \Delta(t, h_{ji} - \min_i)), \\ h_{ji} - \Delta(t, h_{ji} - \min_i), & \text{якщо } f(H_j + \Delta(t, \max_i - h_{ji})) < f(H_j - \Delta(t, h_{ji} - \min_i)), \end{cases}$$

де $\Delta(t, y) = y \cdot w_i(t)$; $w_i(t)$ – коефіцієнт, що залежить від відношення t/T.

Наприклад, коефіцієнт $w_i(t)$ може бути заданий формулою:

$$w_i(t) = r\left(1 - \frac{t}{T}\right)^{k_i},$$

де r = rand[0; 1] – випадково згенероване число в інтервалі [0; 1];

$k_i > 0$ – параметр, що задає користувач.

2. ***Випадкова мутація*** обраного гена h_{ji} полягає в зміні його значення на величину Δh_{ij}, розраховану за формулою:

$$\Delta h_{ij} = \text{rand}[\min_i \cdot r \cdot q(t); \max_i \cdot r \cdot q(t)],$$

де rand[a; b] – випадково згенероване число в інтервалі [a; b];

$\min_i$ і $\max_i$ – мінімальне й максимальне значення i-го гену;

r = rand[0; 1] – випадково згенероване число в інтервалі [0; 1];

$q(t) = \frac{\ln T - \ln t}{\ln T}$.

3. ***Гауссовська (нормальна) мутація*** до обраного для мутації i-го гену j-ої хромосоми h_{ji} застосовується за формулою:

$h_{ij}' = h_{ij} + \varepsilon$,

де ε – випадкове число, отримане за нормальним розподілом (Коші, або будь-якому іншому розподілу) з нульовим середнім і $\sigma_i = \frac{T - t}{T} \cdot \frac{\max_i - \min_i}{3}$.

Число ε може бути додане до одного гена. Можливий варіант додавання випадкового вектора до всієї хромосоми.

4. ***Мутація*** обраного гена h_{ij} ***на основі квадратичної апроксимації***.

Крок 1. Обчислити значення фітнесс-функції при $h_{ij} + \Delta h_{ij}$ і при $h_{ij} - \Delta h_{ij}$: $f(h_{ij} + \Delta h_{ij})$ і $f(h_{ij} - \Delta h_{ij})$. Значення Δh_{ij} можуть бути обчислені за формулами знаходження Δ і ε, аналогічними нерівномірній та Гауссовській мутації.

Крок 2. Апроксимувати точки h_{ij}, $h_{ij} + \Delta h_{ij}$ і $h_{ij} - \Delta h_{ij}$ у параболу.

Крок 3. Знайти мінімальне значення отриманої кривій $f_{\text{параб min}}$ і відповідне значення точки в просторі ознак, що відповідає мінімальному значенню параболи $h_{ij\ \text{min}}$.

Крок 4. Присвоїти: $h_{ij} = h_{ij\ \text{min}}$.

Важливо відзначити, що описані оператори мутації можуть застосовуватися й для бінарних хромосом, попередньо перетворених до реальних числових значень із погляду розв'язуваної задачі. Після застосування описаних вище операторів мутації для таких хромосом їх необхідно знову перетворити до бінарного вигляду, застосувавши використовуваний метод кодування.

Формування нового покоління

Після схрещування та мутації необхідно створити нову популяцію. Види операторів формування нового покоління (репродукції, редукції) практично збігаються з видами операторів відбору батьків, що передбачають формування проміжного масиву особин, допущених до схрещування.

При формуванні нового покоління необхідно вирішити проблему: які з нових особин увійдуть у наступне покоління, а які – ні. Для цього застосовують один із двох способів.

Перший спосіб полягає в тому, що нові особини (нащадки) займають місця своїх батьків. Після чого починається наступний етап, у якому нащадки оцінюються, відбираються, дають потомство й поступаються місцем своїм нащадкам.

Недоліком даного способу є можливість втрати найбільш пристосованої особини попереднього покоління. Одним зі способів вирішення даної проблеми може бути використання *принципу "елітизму"*, що полягає в тому, що особини з найбільшою пристосованістю гарантовано переходять у нову популяцію. Їхнє число може бути від 1 і більше. Кількість елітних особин *KI*, які гарантовано перейдуть у наступну популяцію, може бути обчислена за формулою:

$KI = (1 - S_O) \cdot N,$

де S_O – ступінь відновлення популяції, що перебуває в діапазоні [0,95;1,0]; N – розмір популяції.

Використання принципу “елітизму” дозволяє прискорити збіжність генетичного методу. Недолік використання даної стратегії в тому, що підвищується ймовірність попадання методу в локальний мінімум.

Другий спосіб заснований на тому, що створюється проміжна популяція, яка містить у собі як батьків, так і їхніх нащадків. Члени цієї популяції оцінюються, а потім з них вибираються N найкращих, які й увійдуть у наступне покоління.

Другий варіант є більше оптимальним, але він вимагає сортування масиву розміром $2N$.

Другий варіант формування нового покоління можна реалізувати за допомогою *принципу витиснення*, що носить двохкритеріальний характер – те, чи буде особина з репродукційної групи заноситися в популяцію нового покоління, визначається не тільки величиною її пристосованості, але й тим, чи є вже у популяції наступного покоління особина з аналогічним хромосомним набором. Із всіх особин з однаковими генотипами перевага спочатку віддається тим, чия пристосованість вище. Таким чином, досягаються дві мети: по-перше, не губляться кращі знайдені рішення з різними хромосомними наборами, а по-друге, у популяції постійно підтримується достатня генетична розмаїтість. Витиснення в цьому випадку формує нову популяцію скоріше з далеко розташованих особин, замість особин, що групуються біля поточного знайденого рішення.

Критерії зупинення

Одним з важливих етапів у генетичних методах є визначення критеріїв зупину. Очевидно, еволюція – нескінченний процес, у ході якого пристосованість особин поступово підвищується. Примусово зупинивши цей процес через досить довгий час після його початку й обравши найбільш пристосовану особину в

поточному поколінні, можна одержати не абсолютно точну, але близьку до оптимальної відповідь.

Як правило, в якості критерію зупинення застосовується *обмеження на максимальну кількість ітерацій* функціонування методу (тобто обмеження на кількість поколінь). Кількість популяцій може бути будь-якої, але частіше за все обирають 50-100 популяцій. Якщо в якості критерію зупину обирається максимальна кількість ітерацій, то задається кількість ітерацій T, в результаті чого цикл генетичного пошуку виконується T раз.

Зупинення роботи генетичного методу може відбутися також у випадку, якщо *популяція вироджується*, тобто якщо практично немає розмаїтості в генах особин популяції. Виродження популяції називають передчасною збіжністю.

Якщо в якості критерію зупиннення обирається виродження популяції, то задаються кількість ітерацій T і відсоток поліпшення значення кращої хромосоми F.

Починаючи із циклу $T+1$, у генетичних операторах обчислюються значення цільової функції для кожної хромосоми останньої популяції, і вибирається із цих значень найкраще значення цільової функції $f_{best_{T+t}}$. З попередніх T поколінь вибирається найкраще значення цільової функції $f_{best_{T+t-1}}$ Після чого обчислюється відсоток поліпшення F' по формулі:

$$F' = \frac{100 \cdot (f_{best_{T+t}} - f_{best_{T+t-1}})}{f_{best_{T+t-1}}}.$$

Якщо значення F' виявляється меншим, ніж F, то критерій зупинення вважається досягнутим, в противному випадку виконується наступний цикл генетичного пошуку.

Прийнятне значення цільової функції $f_{\text{п}}$ також може використовуватися в якості критерію зупину. Якщо в процесі функціонування генетичного методу значення цільової функції f деякої особини досягло значення $f_{\text{п}}$ з визначеною заздалегідь заданою точністю ε, то метод зупиняється. При цьому розв'язком задачі є отримане значення цільової функції $f_{\text{п}}$.

Генетичний пошук в пакеті Matlab

Для використання генетичних методів, у середовищі Matlab передбачена функція ga: [x, Fmin] = ga(@fitnessfun, n, options)

Функція ga знаходить мінімум Fmin функції fitnessfun, що має n параметрів, а також вектор x, що мінімізує цільову функцію.

Параметри роботи функції ga задаються в змінній options, що представляє собою наступну структуру:

```
options =
        PopulationType: 'doubleVector'
        PopInitRange: [2x1 double]
        PopulationSize: 20
        EliteCount: 2
        CrossoverFraction: 0.8000
        MigrationDirection: 'forward'
        MigrationInterval: 20
        MigrationFraction: 0.2000
        Generations: 100
        TimeLimit: Inf
        FitnessLimit: -Inf
        StallLimitG: 50
        StallLimitS: 20
        InitialPopulation: []
        InitialScores: []
        PlotInterval: 1
        CreationFcn: @gacreationuniform
        FitnessScalingFcn: @fitscalingrank
        SelectionFcn: @selectionstochunif
        CrossoverFcn: @crossoverscattered
        MutationFcn: @mutationgaussian
        HybridFcn: []
```

Display: 'final'

PlotFcns: []

OutputFcns: []

Vectorized: 'off'

Для зміни значень параметрів функції ga використовується команда gaoptimset, а для одержання поточних параметрів – функція gaoptimget.
Наприклад, для встановлення розміру популяції в 100 особин необхідно використати наступну команду:

```
options = gaoptimset('PopulationSize', 100)
```

Функція gaoptimset дозволяє також передавати параметри в обрані функції, що реалізують основні генетичні оператори. Наприклад, для використання функції мутації mutationgaussian з параметрами scale = 0.5 (середньоквадратичне відхилення) і shrink = 0.75 (параметр скорочення) можна використати таку команду:

```
scale = 0.5; shrink = 0.75;
options = gaoptimset('MutationFcn',{@mutationgaussian,scale,shrink})
```

Приклад

Дано наступні обмеження нерівності і нижні межі

$$\begin{bmatrix} 1 & 1 \\ -1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix},$$
$$x_1 \ge 0, \quad x_2 \ge 0,$$

Наступний код шукає мінімум функції, Fitness_function , яка описана у файлі Fitness_function.m

```
A = [1 1; -1 2; 2 1];
```

```
b = [2; 2; 3];
lb = zeros(2,1);
[x,fval,exitflag] = ga(@Fitness_function,…
2,A,b,[],[],lb)
```

Результат роботи:

```
Optimization terminated: average change in the fitness value less than options.TolFun.

x = 0.6670    1.3340
fval =  -8.2258
exitflag =  1
```

Текст Fitness_function.m:

```
function y = Fitness_function(x)

y = 0.5*x(1)^2+x(2)^2-x(1)*x(2)-2*x(1)-6*x(2);
```

Matlab, починаючи з версії 7.0, містить візуальний інтерфейсний модуль для роботи з генетичними методами – Genetic Algorithm Tool (GAT), що входить до складу бібліотеки Genetic Algorithm and Direct Search Toolbox. Запуск GAT відбувається за командою:Gatool або OPTIMTOOL(ga).

4.3 Порядок виконання роботи

1. Ознайомитися з основними теоретичними відомостями за темою роботи.
2. Вивчити роботу функції ga пакету Matlab.

3. Розробити за допомогою пакету Matlab програмне забезпечення, що реалізує 2 методи генетичного пошуку. Основні генетичні оператори для реалізації генетичних методів обрати з таблиці 1.1 відповідно до варіанту. Інші параметри, необхідні для генетичного пошуку, обрати самостійно. Вибір параметрів обґрунтувати.

4. Виконати тестування розробленого програмного забезпечення за допомогою вирішення задач оптимізації тестових функцій. Тестові функції y_i (не менше п’яти) для виконання тестування програми обрати самостійно.

5. Вибір тестових функцій обґрунтувати.

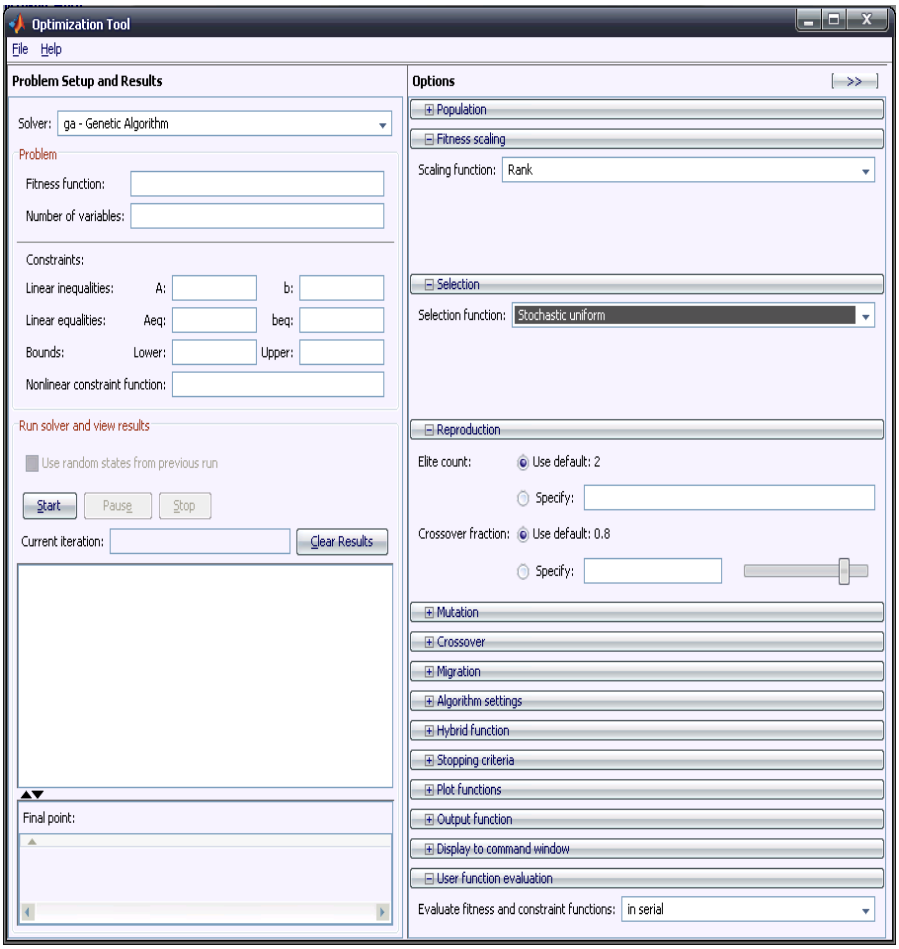


Рис. 4.4 – Візуальний інтерфейсний модуль для роботи з генетичними методами

6. Порівняти одержані результати оптимізації різних функцій за допомогою обох реалізованих генетичних методів. Результати порівняльного аналізу звести до таблиці, попередньо розробивши систему критеріїв порівняння методів генетичного пошуку.

Таблиця 4.1 – Генетичні оператори для виконання завдання

№ варіанту		№ задачі	Генетичні оператори		
			Відбір	Схрещування	Мутація
1		1	пропорційний	однорідне	гауссовська
		2	ранжирування	рівномірне	нерівномірна
2		1	рулетка	арифметичне	проста
		2	пороговий	порівняльне	випадкова
3		1	турнірний	одноточечне	гауссовська
		2	ранжирування	діагональне	нерівномірна
4		1	пропорційний	двохточечне	проста
		2	пороговий	рівномірне	нерівномірна
5		1	рулетка	однорідне	гауссовська
		2	ранжирування	порівняльне	нерівномірна
6		1	турнірний	арифметичне	проста
		2	пороговий	діагональне	випадкова
7		1	пропорційний	одноточечне	гауссовська
		2	ранжирування	рівномірне	випадкова
8		1	рулетка	двохточечне	проста
		2	пороговий	порівняльне	нерівномірна
9		1	турнірний	однорідне	гауссовська

		2	ранжирування	діагональне	випадкова
10		1	пропорційний	арифметичне	проста
		2	пороговий	рівномірне	випадкова

Висновки:

У даній лабораторній роботі:

– Ознайомилися з основними теоретичними відомостями за темою роботи.

– Вивчили роботу функції ga пакету Matlab.

4.4 Зміст звіту

1. Тема та мета роботи.
2. Короткі теоретичні відомості.
3. Текст розробленого програмного забезпечення з коментарями, а також текст програми для тестування розроблених генетичних методів.
4. Графіки та аналітичні вирази обраних тестових функцій.
5. Результати роботи програмного забезпечення (таблиця порівняльного аналізу розроблених генетичних методів).
6. Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

4.5 Контрольні питання

7. Порівняйте методи генетичного пошуку з іншими методами оптимізації.
8. Які методи відносять до генетичних?
9. В чому переваги генетичних методів?

10. Проаналізуйте умови ефективного використання методів генетичного пошуку.
11. Назвіть особливості генетичних методів.
12. Які недоліки генетичного пошуку та в чому вони полягають?
13. Дайте визначення основних термінів, що відносяться до теорії генетичного пошуку: популяція, розмір популяції, число поколінь, хромосома, ген, локус, алель, фенотип, генотип.
14. Проаналізуйте узагальнену схему роботи генетичних методів.
15. Наведіть послідовність виконання узагальненого генетичного пошуку.
16. Які параметри необхідно визначати для роботи генетичних методів?
17. Виконайте порівняльний аналіз канонічних моделей генетичного пошуку.
18. В чому полягають особливості моделі Genitor?
19. Що таке гібридний еволюційний метод? Які існують стратегії взаємодії класичних та генетичних методів?
20. Назвіть відмінності моделі CHC від класичних генетичних методів.
21. Які особливості генетичного методу із змінним часом життя хромосом?
22. Порівняйте мобільний еволюційний метод з класичними методами генетичного пошуку. Для чого призначені оператори CUT та SPLICE?
23. Проаналізуйте паралельні та багаторівневі генетичні методи.
24. Наведіть послідовність виконання генетичного пошуку із зменшенням розміру популяції.
25. Які існують способи кодування параметрів, що оптимізуються, при використанні генетичних методів?
26. Що таке фітнесс-функція?
27. Порівняйте стратегії створення початкової популяції.
28. Виконайте порівняльний аналіз операторів відбору (пропорційний відбір, відбір за допомогою ранжирування, турнірний відбір та відбір з використанням порогу).
29. Які способи формування батьківської пари використовуються в генетичних методах?

30. Проаналізуйте оператори схрещування (*n*-точкове, рівномірне, порівняльне, арифметичне, діагональне).
31. Для чого призначений оператор мутації? Які оператори мутації використовуються в генетичних методах?
32. Яким чином відбувається формування нового покоління?
33. Які критерії зупину використовуються при еволюційному пошуку?
34. Для чого призначена теорема схем? Дайте визначення основних понять, що використовуються в теоремі схем. В чому полягає теорема Холланда про схеми?
35. Порівняйте генетичні стратегії з генетичними алгоритмами та методом імітації відпалу.
36. Виконайте порівняльний аналіз генетичного та генетичного програмування.
37. Проаналізуйте внутрішню структуру функції ga пакету Matlab: основні змінні, параметри, методи та допоміжні функції, їх призначення та використання.
38. Які параметри можна використовувати в функції ga? Яким чином вони задаються? Як отримати поточні параметри функції ga?
39. Проаналізуйте візуальний модуль для роботи з методами генетичного пошуку gatool: призначення, використання, параметри, візуальні компоненти, методи представлення результатів генетичного пошуку.

МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ ТА РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. La Rocca M. Advanced Algorithms and Data Structures / Marcello La Rocca. – Shelter Island: Manning Publications, 2021. – 737 с.
2. Scappini A. The Art of Data Analysis: Non-Technical Skills for Data Analysts / Alberto Scappini., 2018. – 304 с.
3. Weidman S. Deep Learning from Scratch: Building with Python from First Principles / Seth Weidman., 2019. – 235 с.
4. Tunstall L. Natural Language Processing with Transformers: Building Language Applications with Hugging Face / L. Tunstall, L. Werra, T. Wolf., 2022. – 383 с.
5. F. Provost Data Science For Business: What You Need to Know About Data Mining & Data-Analytic Thinking, 2018. – 313 с. – (1st Edition).
6. Aggarwal C. Linear Algebra and Optimization for Machine Learning / Charu C. Aggarwal. – Yorktown: Springer, 2020. – 1087 с. – (1st Edition).
7. J. Zaki M. Data Mining and Machine Learning: Fundamental Concepts and Algorithms / M. J. Zaki, W. Meira, Jr. – Cambridge: Cambridge University Press, 2020. – 776 с. – (2nd Edition).
8. George A. Python Text Mining: Perform Text Processing, Word Embedding, Text Classification and Machine Translation / Alexandra George., 2022. – 320 с. – (1st Edition).
9. Runkler T. Data Analytics: Models and Algorithms for Intelligent Data Analysis / Thomas A. Runkler. – Munich: Springer, 2020. – 176 с.
10. Xiao P. Artificial Intelligence Programming with Python / Perry Xiao. – New Jersey: John Wiley & Sons, Inc., 2022. – 720 с.
11. Russell S. Artificial Intelligence: A Modern Approach / S. Russell, P. Norvig., 2021. – 1168 с. – (4th Edition).
12. Прикладні задачі інтелектуального аналізу даних (DATA MINING). – К.: КНУ ім. Тараса Шевченко, 2018. – 152 с.

13. Buontempo F. Genetic Algorithms and Machine Learning for Programmers: Create AI Models and Evolve Solutions / Frances Buontempo., 2019. – 218 c.

14. Osinga D. Deep Learning Cookbook: Practical Recipes to Get Started Quickly / Douwe Osinga., 2018. – 251 c. – (1st Edition).

15. Wirsansky E. Hands-On Genetic Algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems / Eyal Wirsansky. – Birmingham: Packt Publishing Ltd., 2020. – (348).

16. Berman J. Principles and Practice of Big Data / Jules J. Berman., 2018. – 480 c. – (2nd Edition).