МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра прикладної математики та моделювання складних систем

Допущено до захисту
Завідувач кафедри ПМ та МСС
_____ Коплик І.В.
(підпис)
«___» _____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
спеціальність 113 «Прикладна математика»
освітньо-професійна програма «Прикладна математика»

тема роботи: «**АНАЛІЗ ЯКОСТІ РЕГРЕСІЙНИХ МОДЕЛЕЙ НА ПРИКЛАДІ ЗАДАЧІ КРЕДИТНОГО СКОРИНГУ**»

Виконавець
студент факультету ЕлІТ
Пороскун Олена Олегівна _____
(підпис)

Науковий керівник
док. фіз.-мат. наук, професор
Лисенко Олександр Володимирович _____
(підпис)

Суми – 2022

РЕФЕРАТ

Кваліфікаційна робота: 64 с., 20 рисунків, 1 табл., 21 джерел.

Мета роботи: математичне та комп'ютерне моделювання регресійних лінійних та нелінійних моделей на основі задачі кредитного скорингу; аналіз якості побудованих моделей; виявлення найвагоміших факторних ознак регресійних моделей.

Об'єкт дослідження: дані позичальників кредитних установ, алгоритми для задачі кредитного скорингу.

Предмет дослідження: якість алгоритмів для задачі оцінювання кредитоспроможності.

Методи навчання: стандартні чисельно-аналітичні методи, логістична регресія.

У роботі в рамках задачі кредитного скорингу проведено аналіз регресійних моделей: лінійних та нелінійних. Оглянуто та розв'язано кілька модельних задач, порівняно результати роботи алгоритмів. З'ясовано, що лінійні моделі працюють краще за нелінійні. Також було проаналізувано вплив факторних ознак на кредитоспроможність. Завдяки високій якості моделей стає можливим їх використання для задач скорингу.

Ключові слова: РЕГРЕСІЯ, РЕГРЕСІЙНА МОДЕЛЬ, ЛОГІСТИЧНА РЕГРЕСІЯ, МІРИ ЯКОСТІ, ГРАДІЄНТНИЙ СПУСК.

ЗМІСТ

ВСТУП

Регресійний аналіз – алгоритм для прогнозування результату змінної та ідентифікації змінних (незалежних змінних), які сприяють змінній результату або залежать від неї (цільової або залежної змінної). Простіше кажучи, це техніка пошуку зв’язку між незалежними та залежними змінними для отримання результату. Використовувати та інтерпретувати результат просто. Існує багато типів методів регресії, які широко застосовуються в різних секторах. Деякі з прикладів регресії - передбачити зарплату працівника або дохід компанії за рік [1].

Основним завданням цього алгоритму є визначення впливу факторних ознак (предикторів) на результативний показник. Насамперед для цього необхідно підібрати рівняння зв'язку, що відповідає характеру аналітичної залежності між ознаками, які досліджуються. Рівняння регресії показує як саме змінюється результативна ознака (Y_x) під впливом зміни факторів (x_i).

У загальному вигляді рівняння регресії можна представити так:

$$Y_x = f(x_1, x_2, \dots, x_n,),$$

де Y_x – залежна змінна величина; x – незалежні змінні величини (ознаки).

Залежно від кількості змінних величин виділяють різні види регресійного аналізу, зокрема: парний (простий) аналіз і багатофакторний аналіз (на основі множинної регресії).

Аналіз на основі множинної регресії – вид регресійного аналізу, у якому використовуються більше ніж одна незалежна змінна. Ним користуються прогнозуючи попит. Крім того, спочатку ідентифікуються фактори, що визначають попит, потім визначаються наявні між ними зв'язки та прогнозуються майбутні їх значення. На основі отриманих даних виводиться прогнозоване значення попиту.

Рівняння множинної регресії при лінійній залежності має такий вигляд:

$$Y_{x_1-x_n} = a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n,$$

де a_0 – вільний член рівняння регресії;

$a_1, a_2, \ldots, a_n$ – коефіцієнти регресії або ваги ознак;

$x_1, x_2, \ldots, x_n$ – незалежні змінні (факторні ознаки);

n – кількість незалежних змінних.

Щоб визначити параметри багатофакторної регресії потрібно провести складні розрахунки, використовуючи комп'ютерні інформаційні системи. Одержані результати будуть достовірними і можуть набути широкого застосовування в економічній та іншій діяльності передусім для складання довгострокових прогнозів. Відомо, що однофакторна модель є придатною для короткострокових прогнозів (на 2–3 роки).

Метод регресійного аналізу найбільш результативний за умови здійснення розрахунків завдяки сучасним інформаційним технологіям і системам [2].

Логістична регресія – класичний інструмент для розв’язань завдань регресії та класифікації. Вона набула поширення в скорингу для розрахунку рейтингу позичальників і управління кредитними ризиками. Тому, попри свій зв'язок зі статистикою, логістичну регресію і ROC-аналіз часто можна знайти в наборі Data Mining алгоритмів [3 – 6].

Кредитний скоринг – це метод оцінки кредитного ризику заявок на позику. Використовуючи дані позичальників та статистичні методи, кредитний скоринг намагається виділити вплив різних характеристик заявника на прострочення та неплатежі. Метод дає «оцінку», яку банк може використовувати для ранжування своїх претендентів на кредит з точки зору ризику. Щоб побудувати скорингову модель, розробники аналізують дані про результати раніше наданих позик. Так для оцінки якості моделі визначають

корисні характеристики позичальника. Добре розроблена модель повинна давати більшу частку високих балів позичальникам з позиками, які будуть добре працювати, а також більше частку низьких показників позичальникам, чиї позики не будуть ефективними. Але жодна модель не є ідеальною, і деякі погані облікові записи отримають вищі оцінки, ніж деякі хороші облікові записи [7].

РОЗДІЛ 1
МОДЕЛІ В ЗАДАЧАХ РЕГРЕСІЇ (ОГЛЯД ЛІТЕРАТУРИ)

1.1. Лінійні моделі в задачах регресії

В цьому розділі буде детально розглянуто як лінійні так і нелінійні моделі в задачах регресії, як їх навчати, які проблеми виникають при використанні даних моделей та як оцінити їх якість.

Модель лінійної регресії – часто використовувана в статистиці модель лінійної залежності однієї змінної y від іншої, незалежної змінної x, або декількох змінних. У термінах машинного навчання y - це відповідь, а велика кількість незалежних змінних – ознаки [8].

Уведемо наступні позначення:

- $x = (x^1, \ldots, x^d)$ — опис факторних ознак об'єкта,
- y — результативна ознака,
- $X = (x_i, y_i)\ _{i=1}^{l}$ — навчальна вибірка,
- $a(x)$ — модель,
- $Q(a, X)$ — функціонал помилки алгоритму a на вибірці X,
- навчання: $a(x) = arg \min\limits_{a \in A} Q(a, X)$.

Для розв'язання задачі регресіїї, потрібно задати наступне:

1. Функціонал помилки Q: спосіб оцінки того, добре чи погано працює алгоритм на конкретній вибірці.

2. Сімейство алгоритмів A: який вигляд мають алгоритми, з яких обирається кращий.

3. Метод навчання: як обрати оптимальний алгоритм з сімейства алгоритмів [9].

1.1.1. Опис моделі

Розберемо, який вигляд має сімейство алгоритмів у разі лінійних моделей. Лінійний алгоритм у задачах регресії записується формулою:

$$a(x) = w_0 + \sum_{j=1}^{m} w_j x^j \tag{1.1}$$

де w_0 – вільний коефіцієнт, x^j – факторні ознаки, а w_j - їх ваги.

Якщо додати $(m + 1)$ ознаку, що на кожному об'єкті приймає значення 1, попередню формулу можна записати більш компактно наступним чином:

$$a(x) = \sum_{j=1}^{m+1} w_j x^j = \langle w, x \rangle \tag{1.2}$$

де $\langle w, x \rangle$ – скалярний добуток двох векторів.

В якості міри похибки не можна брати відхилення від прогнозу $Q(a, y) = a(x) - y$, бо тоді не буде мінімуму функціоналу при істинній відповіді $a(x) = y$.

Найбільш простий спосіб – взяти за похибку відхилення $|a(x) - y|$. Однак функція модуля не є гладкою, і для оптимізації такого функціоналу важко застосувати градієнтні методи. Тому в якості міри обирають квадрат відхилення $(a(x) - y)^2$.

Функціонал помилки, що називається середньоквадратичною помилкою алгоритму, запишемо формулою:

$$Q(a, x) = \frac{1}{l} \sum_{i=1}^{l} (a(x_i) - y_i)^2 \tag{1.3}$$

Вираз (1.3) подамо у вигляді функції похибок:

$$Q(w,x) = \frac{1}{l}\sum_{i=1}^{l}(\langle w, x_i\rangle - y_i)^2 \tag{1.4}$$

1.1.2. Навчання моделі

Далі опрацюємо навчання моделі лінійної регресії, тобто знаходження її параметрів. Для цього раніше було уведено вираз для функції помилок (1.4), який потрібно мінімізувати:

$$Q(w,x) = \frac{1}{l}\sum_{i=1}^{l}(\langle w, x_i\rangle - y_i)^2 \to \min_{w} \tag{1.5}$$

Згідно з виразом (1.2) в число ознак входить також постійна ознака, тому можна відкинути постійну складову в попередній формулі [10].

1.1.3. Градієнтний спуск для лінійної регресії

У випадку парної регресії ознака всього одна, а лінійна модель виглядає наступним чином [11-13]:

$$a(x) = w_i x + w_0 \tag{1.6}$$

де w_i і w_0 — два параметри.

Середньоквадратична помилка приймає вигляд:

$$Q(w_0, w_1, X) = \frac{1}{l}\sum_{i=1}^{l}(w_1 x_i + w_0 - y_i)^2 \tag{1.7}$$

Для знаходження оптимальних параметрів буде застосовуватися метод градієнтного спуска. Щоб це зробити, необхідно спочатку обчислити частинні похідні функції похибки:

$$\begin{aligned} \frac{\partial Q}{\partial w_1} &= \frac{2}{l}\sum_{i=1}^{l}(w_1 x_i + w_0 - y_i)x_i, \\ \frac{\partial Q}{\partial w_0} &= \frac{2}{l}\sum_{i=1}^{l}(w_1 x_i + w_0 - y_i) \end{aligned} \tag{1.8}$$

Дуже важливо при використанні методу градієнтного спуска правильно підбирати крок. Конкретних правил підбору кроку не існує, але є декілька корисних закономірностей.

Якщо довжина кроку занадто мала, то метод буде не поспішаючи, але вірно крокувати у бік мінімуму. Якщо ж взяти занадто велику довжину кроку, тоді з'являється ризик, що метод буде перестрибувати через мінімум. Більше того, є ризик того, що градієнтний спуск не зійдеться.

Має сенс використовувати змінний розмір кроку: спочатку, коли точка мінімуму знаходиться далеко, рухатися швидко, а пізніше, через деяку кількість ітерації — робити більше акуратні кроки. Один зі способів задати розмір кроку наступний:

$$\eta_t = \frac{k}{t} \tag{1.9}$$

де k — константа, яку необхідно підібрати, а t — номер кроку.

У випадку багатомірної лінійної регресії використовується той же самий підхід — необхідно вирішувати задачу мінімізації:

$$Q(w, X) = \frac{1}{l} \|Xw - y\|^2 \to \min_w \tag{1.10}$$

де $\|x\|$ — норма вектора x. Формула для обчислення градієнта приймає наступний вигляд:

$$\nabla_w Q(w, X) = \frac{2}{l} X^T (Xw - y) \tag{1.11}$$

Варто відзначити, що вектор $Xw - y$, що є присутнім у цьому виразі, є вектором помилок.

1.1.4. Стохастичний градієнтний спуск

Ідея стохастичного градієнтного спуска заснована на тому, що в сумі у виразі для j-компоненти градієнта i-й доданок вказує на те, як потрібно поміняти вагу w_j, щоб якість збільшилася для i-го об'єкта вибірки. Вся сума за такої умови задає, як потрібно змінити цю вагу, щоб підвищити якість для всіх об'єктів вибірки. У стохастичному методі градієнтного спуска градієнт функції якості обчислюється тільки на одному випадково обраному об'єкті навчальної вибірки. Це дозволяє обійти вищезгаданий недолік звичайного градієнтного спуска.

Отже, алгоритм стохастичного градієнтного спуска наступний. Спочатку вибирається початкове наближення $w^0 = 0$.

Далі послідовно обчислюються ітерації w^t: спочатку випадковим образом вибирається об'єкт x_i з навчальної вибірки X і обчислюється вектор градієнта

функції якості на цьому об'єкті, а наступне наближення виходить із попереднім вирахуванням помноженого на крок η_t отриманого вектора:

$$w^t = w^{t-1} - \eta_t \nabla Q(w^{t-1}, \{x_i\}) \tag{1.12}$$

Ітерації припиняються при досягненні певної умови, наприклад:

$$\|w^t - w^{t-1}\| < \varepsilon \tag{1.13}$$

Стохастичний градієнтний спуск має цілий ряд переваг. По-перше, кожний крок виконується значно швидше кроку звичайного градієнтного методу, а також не потрібно постійно зберігати всю навчальну вибірку в пам'яті. Це дозволяє використовувати для навчання вибірки настільки більші, що вони не містяться у пам’яті комп'ютера. Стохастичний градієнтний спуск також можна використовувати для онлайн-навчання, тобто в ситуації, коли на кожному кроці алгоритм одержує тільки один об'єкт і повинен урахувати його для корекції моделі [11-13].

1.1.5. Функції втрат

Для того, щоб визначити якість алгоритму, потрібно здійснити обчислення частки неправильних відповідей для об’єктів навчальної вибірки [11, 16]:

$$Q(a, x) = \frac{1}{l} \sum_{i=1}^{l} [a(x_i) \neq y_i] \tag{1.14}$$

Варто ввести таке важливе поняття в задачах лінійної класифікації як поняття відступу:

$$M_i = y_i\langle w, x_i\rangle \tag{1.15}$$

Відступ є величиною, що визначає коректність відповіді. Якщо відступ більше нуля, то класифікатор дає вірну відповідь для такого об'єкта, в іншому випадку — помиляється.

Завдяки цій величині можна записати вираз (1.14) наступною формулою:

$$Q(a,x) = \frac{1}{l}\sum_{i=1}^{l}[y_i\langle w, x_i\rangle < 0] = \frac{1}{l}\sum_{i=1}^{l}[M_i < 0] \tag{1.16}$$

Функція втрат – це та функція, яка стоїть під знаком суми. В цьому випадку це гранична функція втрат, графік якої в залежності від відступу показано на рис. 1.1:

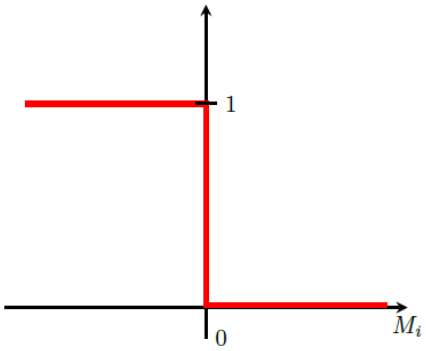


Рисунок 1.1 – Графік граничної функції втрат

У точці 0 ця функція розривна, що призводить до того, що не можна використати метод градієнтного спуску.

Взявши яку-небудь гладку оцінку граничної функції:

$$[M_i < 0] \leq \check{L}(M_i) \tag{1.17}$$

стає можливим вивести формулу для оцінки $\check{Q}(\mathrm{a}, \mathrm{X})$ функціоналу помилки $Q(\mathrm{a}, \mathrm{X})$:

$$Q(a, x) \leq \check{Q}(\mathrm{a}, \mathrm{X}) = \frac{1}{l} \sum_{i=1}^{l} \check{L}(M_i) \tag{1.18}$$

Тепер мінімізувати потрібно буде не вираз (1.14), а іншу функцію, яка є верхньою оцінкою:

$$Q(a, x) \leq \check{Q}(\mathrm{a}, \mathrm{X}) = \frac{1}{l} \sum_{i=1}^{l} \check{L}(M_i) \to \min_{a} \tag{1.19}$$

Припускається, що в точці мінімуму цієї оцінки зверху число похибок буде мінімальним. Взагалі, це не завжди так.

Приклади подібних функцій зображені на рис. 1.2. Зокрема, це наступні функції втрат:

– логістична (логістична регресія):

$$\check{L}(M) = \log_2(1 + \exp(-M)) \tag{1.20}$$

– кусково-лінійна (метод опорних векторів):

$$\check{L}(M) = max(0, 1 - M) \tag{1.21}$$

– експонентна:

$$\check{L}(M) = exp(-M) \tag{1.22}$$

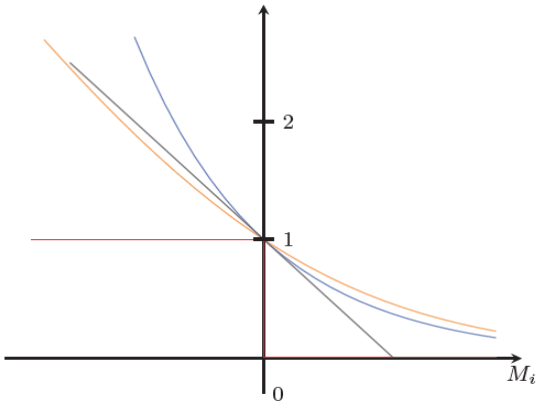


Рисунок 1.2 – Графіки функцій втрат: гранична (червоний колір), логістична (жовтогарячий колір), кусково-лінійна (сірий колір), експонентна (синій колір).

Якщо взяти логістичну функцію втрат, то функціонал похибки буде виражатися формулою (1.23).

$$\check{Q}(a,X) = \frac{1}{l}\sum_{i=1}^{l}\left(1 + \ln(\exp(-M_i))\right) = \\ = \frac{1}{l}\sum_{i=1}^{l}\left(1 + \ln\left(\exp(-y_i\langle w, x_i\rangle)\right)\right) \tag{1.23}$$

Цей вираз є гладким, тому можемо застосувати метод градієнтного спуска.

Зауважимо, що у випадку, коли число похибок буде дорівнювати нулю, однаково в процесі навчання алгоритму відступи будуть ставати більшими, тобто достовірність одержаних результатів збільшуватиметься.

1.1.6. Перенавчання. Регуляризація

Допустимо, що під час вирішення задачі класифікації був побудований алгоритм, на якому частка помилок навчальної вибірки дорівнювала 0.2, і вона є прийнятною.

Однак так як алгоритм не можна узагальнити, то немає впевненості в тому, що така ж частка похибок буде для нових даних. Можливий випадок, що вона набуде значення 0.9. Це пояснюється тим, що алгоритм не зміг знайти закономірності навчальних об'єктів й використати їх, щоб класифікувати нові об'єкти, тобто він не узагальнюється для всієї вибірки даних. А проте модель якимось чином підлаштувалась під навчальні дані й видала гарні результати при навчанні без знаходження справедливої логіки. Це і є випадком перенавчання моделі.

Краще зрозуміти цю проблему можна на такому прикладі. На наступному графіку зображена справжня залежність та об'єкти навчальної вибірки:

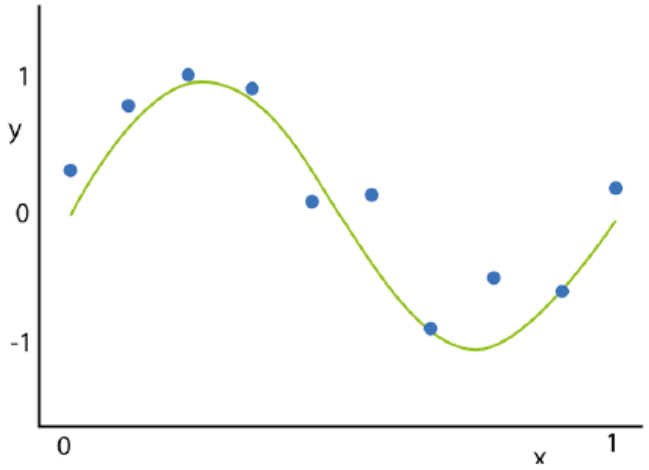


Рис. 1.3. Істинна залежність (зелена лінія) і елементи навчальної вибірки (зображені синіми точками).

Видно, що справжня залежність є нелінійною й має два екстремуми. Якщо сімейство алгоритмів - множина багаточленів 3-го ступеня:

$$a(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3,$$

то після навчання отримана крива буде досить добре описувати й навчальну вибірку, і істину залежність.

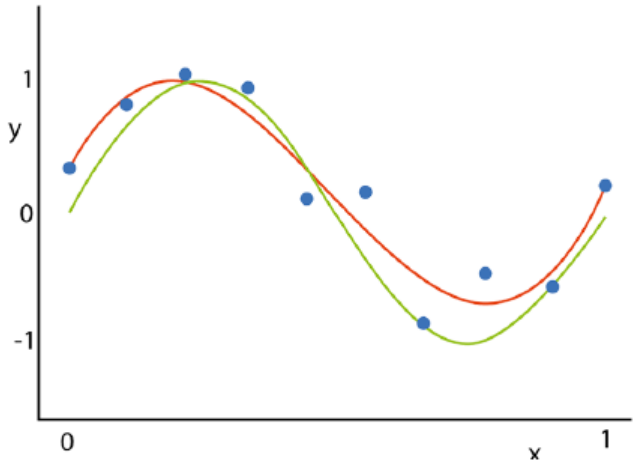


Рис. 1.4. Модель $a(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$.

У цьому випадку якість алгоритму гарна, але немає ідеального збігу. Постає питання, а чи можна домогтися збігу збільшенням складності алгоритму.

При використанні багаточленів 9-ой ступеня вже має місце перенавчання.

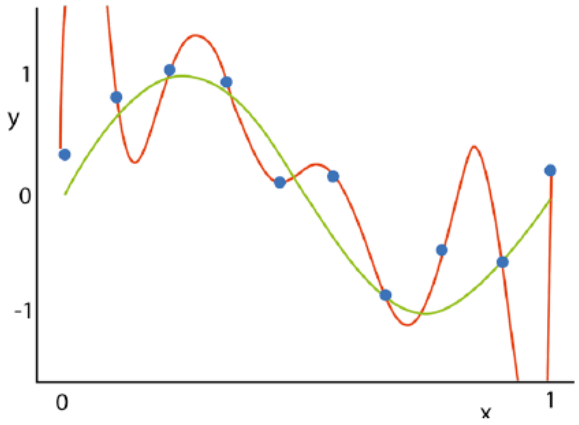


Рис. 1.5. Модель $a(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_9 x^9$.

Відновлена залежність дає ідеальні відповіді на всіх об'єктах навчальної вибірки, але водночас в будь-якій іншій точці сильно відрізняється від істинної залежності. Така ситуація називається перенавчанням. Алгоритм занадто сильно налаштувався під навчальну вибірку ціною того, що він буде давати погані відповіді на нових точках.

Отже, у випадку перенавчання, навчальні дані характеризуються добре, а нові – ні. Виявити перенавчання, за допомогою тільки навчальної вибірки, нереально, бо як і переучений, так і досить гарно навчений алгоритми опишуть її вдало. Тому варто використати додаткові дані.

Є кілька способів, щоб виявити цю проблему. По-перше, відкладена вибірка – коли на деякій частині даних ми навчаємо алгоритм, на решті – перевіряємо його. По-друге, крос-валідація, трішки складніший метод відкладеної вибірки. По-третє, використовувати міри якості моделі.

Далі розкриємо тему регуляризації як способу боротьби з перенавчанням у лінійних моделях.

Якщо ваги в моделі мають великі значення, то є висока ймовірність перенавчання. Для вирішення подібної ситуаціїї мінімізується вираз для оновленого функціоналу похибки $Q(a, X)$ з регуляризатором. Квадратичний регуляризатор – найпростіший. Він має вигляд:

$$\|w\|^2 = \sum_{j=1}^{d} w_j^2 \tag{1.24}$$

Тут виникає задача оптимізації записана формулою (1.25).

$$Q(w, X) + \lambda\|w\|^2 \to \min_w \tag{1.25}$$

Отже, коли буде навчатися модель, тоді буде враховуватися те, що не варто мати дуже великі ваги факторних ознак.

У виразі (1.25) з’явився коефіцієнт λ, його будемо називати коефіцієнтом регуляризації. Чим він більше, тим менша складність алгортиму. Також з досить низькими значеннями виникає велика вірогідність перенавчання, тобто

модель стає занадто складною. Через те необхідно знайти оптимальне значення цього параметру. Його переважно підбирають на кросвалідації.

Для кращого розуміння сенсу регуляризації варто дослідити задачу умовної оптимізації:

$$\begin{cases} Q(w, X) \to \min\limits_{w} \\ \|w\|^2 \leq \mathrm{C} \end{cases} \tag{1.26}$$

Додавання регуляризатора вимагає пошуку розв’язку задачі мінімізації в певній круглій області з центром в нулі (рис. 1.6).

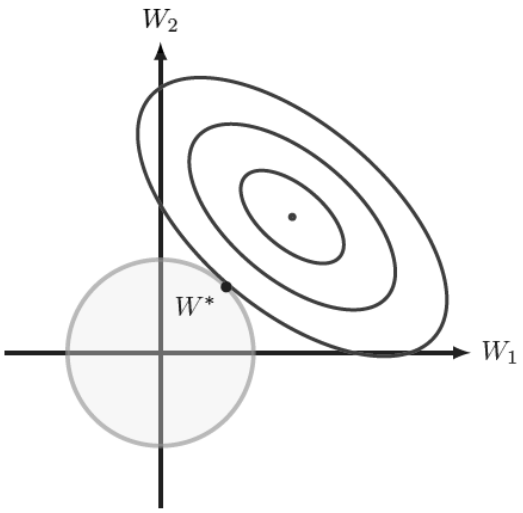


Рисунок 1.6 – Геометричний сенс умовної регуляризації. Точка в центрі овалу - оптимум функції, овальні лінії - лінії рівня функції, точка дотику кола та овалу - оптимум функції при уведеному обмеженні.

Отже, розв’язок задачі з регуляризатором не буде описуватися дуже великими вагами.

Описаний вище квадратичний регуляризатор ($L_2 -$ регуляризатор) є гладким і опуклим, тому можна застосувати градієнтний спуск. Також існує $L_1 -$ регуляризатор:

$$\|w\|_1 = \sum_{j=1}^{d} |w_j| \tag{1.27}$$

Це L_1 – норма вектору вагів, він не є гладким. Якщо взяти цей регуляризатор, то значення деяких вагів стануть рівними нулю. Тобто такий регуляризатор дозволяє застосувати в алгоритмі тільки найбільш важливі ознаки.

1.1.7. Оцінка якості моделей. Крос валідація

Оцінимо якість алгоритму завдяки відкладеній вибірці. У цьому випадку варто поділити вибірку на дві частини: у першій буде відбуватися навчання алгоритму, а у другій – тестування, тобто оцінювання якості моделі, зокрема, розрахування частки помилок та інших мір якості залежно від поставленої задачі.

Зазвичай вибірку розбивають у наступних співвідношеннях: 70/30, 80/20 або 0.632/0.368.

Перевага подібного підходу – навчання алгоритму відбувається тільки один раз, однак за такої умови результат сильно залежить від того, як саме поділили дані.

Більш складний підхід — крос валідація. У цьому випадку вибірка ділиться на k блоків з орієнтовно однаковим розміром. Далі послідовно береться кожен блок як тестовий, а всі інші – як навчальні.

Потім будуть отримані k показників якості. У результаті усереднення отримується оцінка якості за крос валідацією.

Здебільшого беруть k = 3, 5, 10. Чим більша кількість блоків, тим більше раз доводиться проводити навчання алгоритму. Тому на великих вибірках варто вибирати невеликі значення k, тому що навіть при видалені 1/3 вибірки даних, тих даних, що залишились, буде досить для навчання.

Насправді найчастіше вибирається k = 10, коли модель навчається на 9/10 даних, і тестується на 1/10. Дослідження показали, що в цьому випадку виходить найдостовірніша оцінка вихідної помилки моделі [9].

1.2. Метрики якості

Метрики якості можна застосовувати для наступних цілей [9]:

1. Знаходження функціонала похибки (при навчанні).
2. Підбір гіперпараметрів (при вимірі якості на крос-валідації).
3. Оцінювання моделі: чи придатна вона для вирішення задачі.

Далі висвітлимо те, якими саме метриками вимірювати якість у задачах класифікації.

1.2.1. Частка правильних відповідей (accuracy)

У задачах класифікації як міру якості беруть частку неправильних відповідей $\frac{1}{l}\sum_{i=1}^{l}[a(x_i) \neq y_i]$.

Втім в задачах класифікації прийнято обирати метрики так, щоб необхідно було їх максимізувати, тоді як в задачах регресії навпаки – мінімізувати. Тому записують цю міру наступним виразом [9, 14]:

$$accuracy(a, X) = \frac{1}{l}\sum_{i=1}^{l}[a(x_i) = y_i] \tag{1.28}$$

1.2.2. Матриця похибок. Точність і повнота

За допомогою так званої матриці помилок (або похибок) досить зручно класифікувати різні випадки, які є співвідношеннями між результатом роботи алгоритму і правильною відповіддю (рис. 1.7) [14].

	$y = 1$	$y = -1$
$a(x) = 1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False Negative (FN)	True Negative (TN)

Рисунок 1.7 – Матриця похибок

Коли алгоритм відносить об'єкт до класу +1, мають на увазі, що алгоритм спрацював. Якщо ж модель спрацювала і об'єкт дійсно відноситься до цього класу, то буде так зване вірне спрацьовування (TP), а якщо об'єкт насправді відноситься до іншого класу, тоді буде хибне спрацьовування (FP).

Якщо алгоритм дає у відповіді –1, кажуть, що він пропустив об'єкт. Якщо відбувся пропуск об'єкта класу +1, то це хибний пропуск (FN). Якщо ж модель пропустила об'єкт класу –1, то буде істинний пропуск (TN).

Отже, є два види похибок: хибні спрацьовування й хибні пропуски.

Далі розглянемо наступні дві метрики. Точність (precision) - метрика, яка показує наскільки можна довіряти класифікатору, коли він спрацьовує:

$$\text{precision}(a, X) = \frac{TP}{TP + FP} \tag{1.29}$$

Повнота (recall) – друга метрика, що показує на якій частці істинних об’єктів першого класу алгоритм має спрацьовування:

$$\text{recall}(a, X) = \frac{TP}{TP + FN} \tag{1.30}$$

1.2.3. F-міра

Для більш показових результатів використаємо гармонійне середнє, або F-міру [14]:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{1.31}$$

Якщо потрібно надати перевагу точності або повноті, варто взяти розширену F-міру, в якої є параметр β:

$$F = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{1.32}$$

Для прикладу, якщо взяти значення параметру $\beta = 0.5$, то повнота стане важливіше, а якщо взяти $\beta = 2$, навпаки – точність.

1.2.4. PR-крива

Розглянемо криву точності-повноти (PR-криву), яка являє собою спосіб оцінки приналежності класу. Повнота відкладається по осі x, а точність – по осі y. Кожна точка кривої буде відповідати класифікатору з окремим значенням порога. У задачах з кількістю об’єктів порядку декількох тисяч або більше, ця крива виглядає так (рис. 1.8):

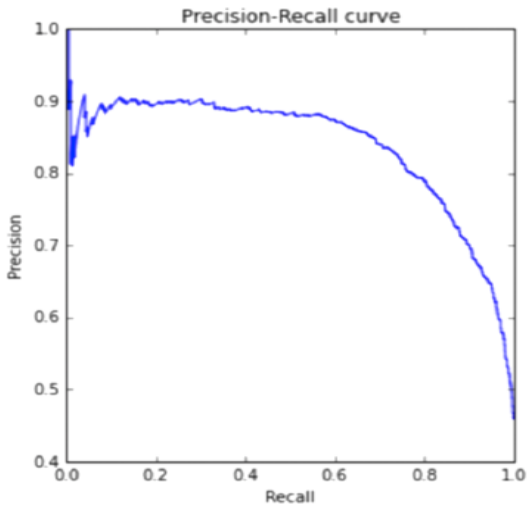


Рисунок 1.8 – PR-крива

Варто додати те, що PR-крива починається з точки $(0;\ 0)$, а закінчується точкою $(1;\ r)$, де $r -$ частка об'єктів першого класу. Коли маємо ідеальний класифікатор при якому існує таке значення порогу, що і точність, і повнота дорівнюють 100%, то це означає, що крива буде проходити через точку $(1;\ 1)$. Отже, чим краще оцінки, тим ближче крива пройде до цієї точки. Площа під цією кривою може бути мірою якості оцінок приналежності до першого класу. Подібна метрика має назву площа під PR-кривою, або AUC-PRC.

1.2.5. ROC-крива

ROC-крива - наступний спосіб виміряти якість оцінок приналежності до першого класу. Вона будується в осях False Positive Rate (вісь x) і True Positive Rate (вісь y), значення яких можна виразити як:

$$FPR = \frac{FP}{FP + TN}, \quad TPR = \frac{TP}{TP + FN} \tag{1.33}$$

Якщо вибірка є великою, тоді ROC-крива виглядає так (рис. 1.9).

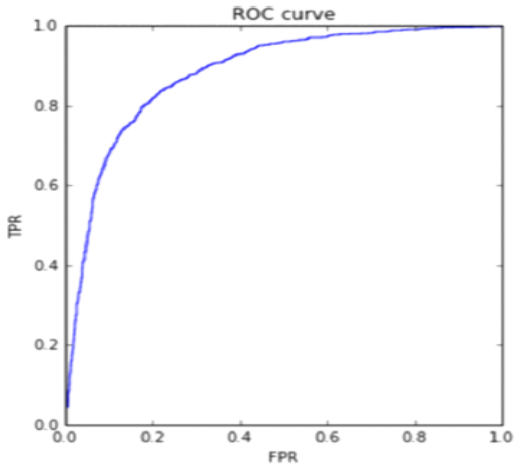


Рисунок 1.9 – Крива ROC в реальних задачах з десятками тисяч об'єктів

Крива починається з точки $(0; 0)$ і приходить в точку $(1; 1)$. Водночас крива повинна пройти через точку $(0; 1)$, якщо існує ідеальний класифікатор. Чим краще будуть оцінки, тим ближче крива буде до цієї точки, а площа під кривою буде характеризувати якість оцінок приналежності до першого класу. Ця метрика має назву площа під ROC-кривою, або AUC-ROC.

1.3. Нелінійна задача регресії

Перш ніж перейти до наступного алгоритму машинного навчання, необхідно обговорити тему нелінійної регресії (nonlinear regression). Часто на практиці відсутня лінійна залежність між вектором ознак і відповідю. Припустимо, що ми вирішуємо завдання оцінки стану об'єкта, що рухається з постійним прискоренням, в залежності від часу руху. Відомо, що в даному випадку між становищем і часом є квадратична залежність і в таком випадку неможливо підібрати коефіцієнти ваг лінійної регресії так, щоб алгоритм давав правдоподібні відповіді - квадратична функція зростає набагато швидше, ніж лінійна.

Звичайно, запропонована задача вирішується абсолютно іншими методами, відмінними від методів машинного навчання, проте вона демонструє проблему, з якою може зустрітися розробник в процесі вирішення завдань: що, якщо потрібно застосувати машинне навчання для відновлення регресії в умовах обмеженості ресурсів як для навчання, так і для використання алгоритму, але лінійна регресія, в першому наближенні, абсолютно не підходить як модель?

В такому випадку можна спробувати застосувати один з методів створення ознак для об'єктів, який полягає в функціональному перетворенні вже існуючих ознак.

Візьмемо все ту ж задачу з положенням об'єкта. Очевидно, що таке рівняння погано описує становище об'єкта в залежності від часу:

$$a(t) = w_1 x + w_0 \tag{1.34}$$

Спробуємо розширити простір ознак наступним перетворенням:

$$\varphi(< x_1, x_2, \dots, x_k >) = (< x_1, x_2, \dots, x_k, x_1^2, x_2^2, \dots, x_k^2 >) \tag{1.35}$$

В такому випадку ми отримаємо іншу модель для нового простору ознак:

$$a(t) = w_2 x_2 + w_1 x_1 + w_0 = w_2 x_1^2 + w_1 x_1 + w_0 \tag{1.36}$$

Але ж це саме те рівняння, коефіцієнти якого і потрібно підбирати. А проте з точки зору набору ознак $< x_1, x_2 >$ – це все та ж лінійна регресія.

Даний метод дуже важливий в процесі переходу від простих лінійних алгоритмів до нелінійних - спочатку необхідно вибрати лінійну модель, а потім доповнити простір ознак зміненими за допомогою нелінійних перетворень вихідними ознаками [13, 17, 21].

РОЗДІЛ 2
МАТЕМАТИЧНІ МОДЕЛІ

Розглянемо математичні моделі регресії на основі задачі скорингу. Візьмемо лінійну та нелінійну модель. В лінійній моделі використаємо 2 алгоритми: на основі логістичної регресії (з логістичною/логарифмічною функцією втрат) і на основі квадратичної функції втрат. Нелінійну модель побудуємо тільки з логістичною регресією.

2.1. **Розв'язок задачі з різними функціями втрат**

Як було показано у розділі 1, щоб навчити модель потрібно мінімізувати вираз оцінки зверху функції втрат (1.19). Це можна зробити підібравши оптимальні значення вагів ознак.

Прикладами подібних оцінок функції втрат є не тільки функції (1.20 – 1.22), але ще й наступні функції [16-18]:

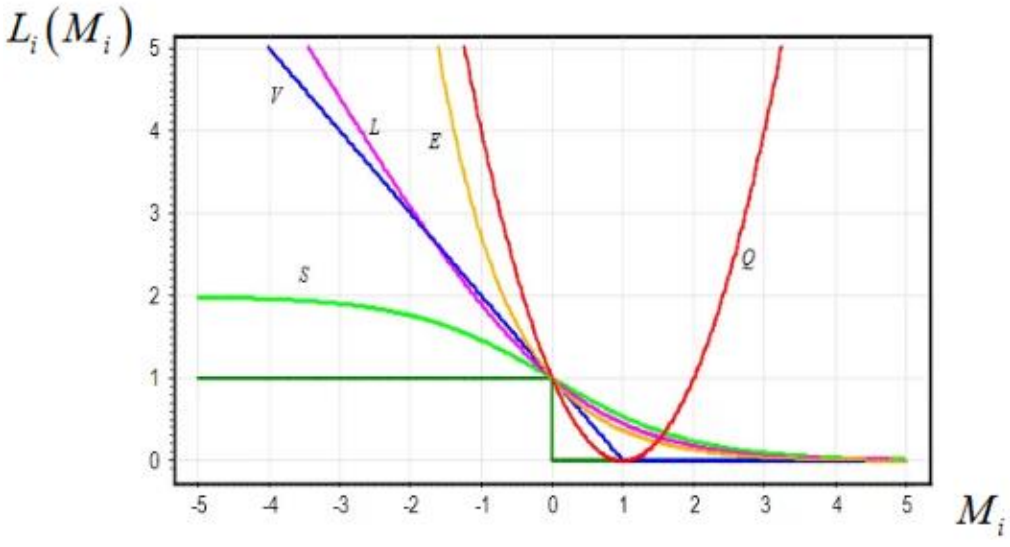


Рисунок 2.1 – Графіки різних функцій втрат: $V(M) = (1 - M)_+$ – кусково-лінійна (SVM); $H(M) = (-M)_+$ – кусково-лінійна (Hebb's rule); $L(M) = \log_2(1 + e^{-M})$ – логарифмічна (LR); $Q(M) = (1 - M)^2$– квадратична

(FLD); $S(M) = 2 \cdot (1 + e^{M})^{-1}$ – сигмоїдна (ANN); $E(M) = e^{-M}$ – експонентна (AdaBoost).

Два з наведених вище в рисунку 2.1 функціонали породжують два різних лінійних класифікатора з різними характеристиками [17]:

- квадратична функція втрат $(1 - M)^2$;
- логарифмічна функція втрат $\log_2(1 + e^{-M})$ – логістична регресія.

Далі будемо розглядати моделі з логарифмічною та квадратичною функцією втрат.

2.1.1. Логістична функція втрат

Розглянемо регресійну модель з логістичною(логарифмічною) функцією втрат), так звану логістичну регресію.

Логістична регресія – це вид множинної регресії, призначенням якої є аналіз зв'язку між декількома незалежними змінними і залежною змінною. Коли залежна змінна є бінарною, використовують бінарну логістичну регресію.

Завдяки цій регресії можна оцінити ймовірність того, що подія настане для певного випробуваного (хворий/здоровий, повернення кредиту/дефолт і т.д.) [15].

Всі регресійні моделі можна записати наступним чином:

$$y = F(x_1, x_2, \dots, x_n,), \tag{2.1}$$

де y – залежна змінна величина; x – незалежні змінні величини (фактори).

У багатофакторній лінійній регресії передбачається, що залежна змінна є лінійною функцією незалежних змінних, тобто:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \quad (2.2)$$

Один із способів використання цієї регресії – оцінювання ймовірності результату події, розрахувавши стандартні коефіцієнти регресії. Зокрема, якщо досліджується результат по позиці, задається змінна y зі значеннями 1 і 0, де 1 означає, що відповідний позичальник розплатився по кредиту, а 0, що відбувся дефолт.

Проте множинна регресія не передбачає, що змінна відгуку приймає тільки два значення (0 і 1), а не значення з множини $(-\infty, \infty)$. Це неодмінно призведе до моделі, яка матиме вихідні значення, що не будуть належати проміжку (0, 1). Але подібні значення не допустимі. Отже, багатофакторна регресія ігнорує обмеження для значень y.

Тому завдання може бути сформульовано по-іншому: замість передбачення бінарної змінної, передбачаємо неперервну змінну зі значеннями на відрізку [0,1]. Цього можна досягти завдяки наступному регресійному рівнянню (логіт – перетворенню), яке має вигляд:

$$P = \frac{1}{1 + e^{-y}}, \quad (2.3)$$

де P – ймовірність події, e — 2.71, основа натуральних логарифмів; y – стандартне рівняння регресії.

Наступний графік показує залежність, що зв'язує ймовірність події і рівняння регресії (рис. 2.2).

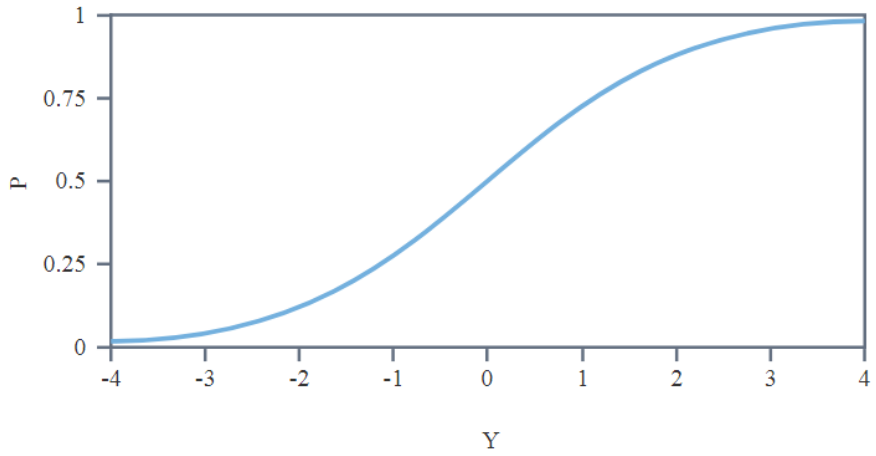


Рис. 2.2 – Графік функції $P(y)$

Завдяки логістичній регресії робиться припущення, що ймовірність настання події p = 1 рівна [11, 19]:

$$\mathbb{P}\{y = 1 \mid x\} = f(z), \tag{2.4}$$

де $z = \theta^T x = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$;

x, θ – вектори-стовбці значень незалежних змінних $(1, x_1, x_2, \ldots, x_n)$ та коефіцієнтів регресії $(\theta_0, \ldots, \theta_n)$, відповідно, а $f(z)$ – сигмоїд-функція (або логіт-функція):

$$f(z) = \frac{1}{1 + e^{-z}}, \tag{2.5}$$

Перехід від формули (2.4) до (2.5) можна розписати завдяки методу максимальної правдоподібності [19, 20]:

$$w = arg\max_{w} P(y \mid x, w), \tag{2.6}$$

де x – ознаки, w – їх ваги, y – простір відповідей.

Розпишемо формулу ймовірності настання події:

$$P(y_1, \dots, y_n | \, x_1, \dots, x_n, w_1, \dots, w_n) = \prod_{i=1}^{n} P(y_i | \, x_i, w) \tag{2.7}$$

Згідно з формулою (1.19) можна записати наступні твердження у яких $a(x)$ — алгоритм:

$$\begin{gathered} \check{L}(M) = L(w) = \prod_{i=1}^{n} P(y_i | \, x_i, w) \\ log \, L(w) = log \, P(y_i | \, x_i, w) \to \max_w \end{gathered} \tag{2.8}$$

$$\begin{gathered} \check{Q}(a, X) = L(x, w) \to \min_w \\ \check{Q}(a, X) = \sum_{i=1}^{n} \log P(y_i | \, x_i, w) \; \to \min_w \end{gathered} \tag{2.9}$$

Так як використовуємо логістичну функцію втрат (1.20), то маємо наступне твердження:

$$\check{L}(M) = \log(1 + e^{-M}) \tag{2.10}$$

Тоді маємо:

$$\begin{gathered} \check{Q}(a, X) = \sum_{i=1}^{n} \log(1 + e^{-M}) = -\sum_{i=1}^{n} \log P(y_i | \, x_i, w) \; \to \min_w \\ \log(1 + e^{-M}) = - \, log \, P(y_i | \, x_i, w) \end{gathered} \tag{2.11}$$

$$P(y_i | \, x_i, w) = \frac{1}{1 + e^{-M}}, \qquad i = 1 \dots n \tag{2.12}$$

Тобто справедливі наступні твердження:

− якщо ймовірність події (клієнт поверне кредит) $f(z) < 0.5$, то подія не відбудеться ($y = 0$),

− якщо ймовірність події (клієнт поверне кредит) $f(z) \geq 0.5$, то подія відбудеться ($y = 1$).

Далі потрібно навчити алгоритм на основі градієнтного спуску [12], щоб знайти коефіцієнти регресії θ(theta) – ваги ознак(w). Навчання полягає в тому, щоб підібрати вагові коефіцієнти так, щоб мінімізувати деяку цільову функцію – логістичну функцію втрат. Маємо наступні перетворення [13]:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(f_i(z_i)) \tag{2.13}$$

де m – кількість елементів у виборці(у нашому випадку кількість клієнтів).

$$Cost(f(z), y) = \begin{cases} -log\big(f(z)\big), & \text{якщо } y = 1 \\ -log\big(1 - f(z)\big), & \text{якщо } y = 0 \end{cases} \tag{2.14}$$

На основі формули (2.5) запишемо ймовірність настання події:

$$P(y_i | \, x_i, \theta) = \ (f(z))^y (1 - f(z))^{1-y} \tag{2.15}$$

Згідно з формулою (2.8) маємо

$$\check{L}(M) = L(\theta) = \prod_{i=1}^{n} P(y_i | \, x_i, \theta) = \prod_{i=1}^{n} (f_i(z_i))^{y_i} (1 - f_i(z_i))^{1-y_i} \tag{2.16}$$

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m} y_i \, log(f_i(z_i)) + (1 - y_i) \, log(1 - f_i(z_i)) \tag{2.17}$$

Тоді для мінімізації маємо домножити на $-\frac{1}{m}$:

$$J(\theta) = -\frac{1}{m} l(\theta) \tag{2.18}$$

Отже, маємо мінімізувати наступну формулу:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y_i \, log(f_i(z_i)) + (1 - y_i) \, log(1 - f_i(z_i))\right] \tag{2.19}$$

Значення градієнту – похідної від функції втрат $J(\theta)$:

$$\theta_j = \theta_j - \alpha \frac{\delta}{\delta\theta} J(\theta),$$
$$\text{де } \frac{\delta}{\delta\theta} J(\theta) = \frac{1}{m}\left[\sum_{i=1}^{m} (f_i(z_i) - y_i) \, x_i\right] \tag{2.20}$$

Підібравши вагові коефіцієнти θ для рівняння регресії знаходимо функцію $f(z)$, яка є ймовірністю того, чи поверне кредит клієнт (наше припущення), згідно з формулами (2.4) та (2.5).

2.1.2. Середньоквадратична функція втрат

Розглянемо регресійну модель з квадратичною функцією втрат. В цьому випадку завдяки перетворенням (1.14 – 1.18) мінімізувати потрібно наступну функцію:

$$Q(a, x) \leq \check{Q}(\mathrm{a}, \mathrm{X}) = \frac{1}{l}\sum_{i=1}^{l} \check{L}(M_i) \to \min_a \,, \tag{2.21}$$
$$\text{де } \check{L}(M) = (1 - M)^2$$

Запишемо даний вираз через базові позначення [18]:

$$Q(a,x) = Q(w) = \sum_{i=1}^{l} (1 - w^T \cdot x_i \cdot y_i)^2 \to \min_w \tag{2.22}$$

де $a(x)$ — алгоритм, x – ознаки, w – їх ваги, y – простір відповідей.

Для знаходження оптимальних параметрів необхідно обчислити частинну похідну функції похибки та зробити наступні перетворення:

$$\begin{gathered} \frac{dQ(w)}{dw} = -2 \sum_{i=1}^{l} (1 - w^T \cdot x_i \cdot y_i) \cdot x_i^T \cdot y_i = 0; \\ \sum_{i=1}^{l} x_i^T \cdot y_i - w^T \sum_{i=1}^{l} x_i \cdot x_i^T \cdot y_i^2 = 0; \\ w^T = \sum_{i=1}^{l} x_i^T \cdot y_i \cdot \left(\sum_{i=1}^{l} x_i \cdot x_i^T \right)^{-1}. \end{gathered} \tag{2.23}$$

РОЗДІЛ 3 РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОГО ЕКСПЕРИМЕНТУ

Для вирішення поставленого завдання використовувався пакет програм MATLAB. Код програм наведений в додатку А, Б, В.

3.1. Опис набору даних

Дана навчальна вибірка — звіт по кредитній історії позичальників фізичних осіб. За результатами характеру і результату погашень всі позичальники розділені на два класи: благонадійний і неблагонадійний, тобто той, поверне кредит і той, хто не зробить цього.

Незалежними змінними є анкетні дані позичальників:

1. Вік.
2. Стать (1 - жіноча, 0 - чоловіча).
3. Сімейний стан: перебуває у шлюбі (1 — так, 0 — ні).
4. Кількість утриманців.
5. Підтверджений сукупний наявний дохід.
6. Досвід роботи, років.
7. Термін проживання в регіоні, років.
8. Ринкова вартість нерухомості у власності, тис. дол.
9. Щомісячний платіж по кредиту, грн.

Залежна змінна (1 — благонадійний, 0 — неблагонадійний позичальник).

Вибірка збалансована – розподіл залежної змінної наступний: 492 благонадійних позичальників з 999.

Для подальшої роботи з даними вибірки їх потрібно стандартизувати. Стандартизація даних полягає в пропорційному масштабуванні даних для

зняття обмежень між даними та перетворення їх у безрозмірні дані для полегшення зважування та порівняння різних індексних даних.

Приклади початкових та стандартизованих даних вибірки можемо побачити на рис. 3.1 та рис. 3.2 для лінійної та нелінійної моделі відповідно.

```
ВХІДНІ НЕСТАНДАРТИЗОВАНІ ДАНІ(лін).
Вхідні дані - вибірка анкетних даних позичальників.
Перші 5 позичальників мають такі анкетні дані.
Назви стовпців:
    1   Вік.    2   Стать.  3   Перебуває_у_шлюбі.  4   Утриманці.  5   Дохід.  6   Досвід_роботи.
    7   Термін_проживання.  8   Нерухомість.    9   Місячний_платіж.    10  Благонадійний_позичальник.
Дані позичальників:

 1      2      3      4      5      6      7      8      9      10

28      1      1      0   9000      9      7      0   3946      0
39      0      1      1  13500     17      6      0   2460      1
31      0      0      2   7000     11      3      0   3126      0
34      0      0      1  10200     15      2     41   3280      1
46      1      1      2   8500     20      8      0   3348      1

ВХІДНІ СТАНДАРТИЗОВАНІ ДАНІ(лін).
Вхідні дані - вибірка анкетних даних позичальників.
Перші 5 позичальників мають такі анкетні дані.
Назви стовпців:
    1   Вік.    2   Стать.  3   Перебуває_у_шлюбі.  4   Утриманці.  5   Дохід.  6   Досвід_роботи.
    7   Термін_проживання.  8   Нерухомість.    9   Місячний_платіж.    10  Благонадійний_позичальник.
Дані позичальників:

-0.90062       1.364     0.83218     -1.0937    -0.57074    -0.62871    -0.59834      -0.756   -0.072414           0
 0.63808    -0.73238     0.83218   -0.042108    0.026602      0.8836    -0.69064      -0.756    -0.96213           1
-0.48098    -0.73238     -1.2005      1.0095    -0.83623    -0.25063    -0.96754      -0.756    -0.56337           0
-0.06133    -0.73238     -1.2005   -0.042108    -0.41145     0.50552     -1.0598       1.351    -0.47117           1
  1.6173       1.364     0.83218      1.0095    -0.63711      1.4507    -0.50604      -0.756    -0.43046           1
```

Рис. 3.1 – Вигляд даних вибірки на прикладі даних 5 клієнтів до стандартизації та після для лінійної моделі

```
ВХІДНІ НЕСТАНДАРТИЗОВАНІ ДАНІ(лін).

Вхідні дані - вибірка анкетних даних позичальників.
Перші 5 позичальників мають такі анкетні дані.
Назви стовпців:
    1   Вік     2   Стать   3   Перебуває_у_шлюбі   4   Утриманці   5   Дохід   6   Досвід_роботи
    7   Термін_проживання   8   Нерухомість     9   Місячний_платіж     10  Благонадійний_позичальник

Дані позичальників:

28      1      1      0   9000      9      7      0   3946      0
39      0      1      1  13500     17      6      0   2460      1
31      0      0      2   7000     11      3      0   3126      0
34      0      0      1  10200     15      2     41   3280      1
46      1      1      2   8500     20      8      0   3348      1
```

```
ВХІДНІ СТАНДАРТИЗОВАНІ ДАНІ(нелін).
Вхідні дані - вибірка анкетних даних позичальників.
Перші 5 позичальників мають такі анкетні дані.
Назви стовпців:
    1   Вік     2   Стать   3   Перебуває_у_шлюбі   4   Утриманці   5   Дохід   6   Досвід_роботи
    7   Термін_проживання   8   Нерухомість     9   Місячний_платіж
Далі йдуть стовпчики перемножені один на одного:
    10 1x1,  11 1x2, ...,  18 1x9 (9 шт),  19 2x2, ...,  26 2x9 (8 шт), ...,  54 9x9(1 шт)
    55  Благонадійний_позичальник

Дані позичальників:

ans =

  Columns 1 through 11

   -0.9006    1.3640    0.8322   -1.0937   -0.5707   -0.6287   -0.5983   -0.7560   -0.0724   -0.1020   -1.2846
    0.6381   -0.7324    0.8322   -0.0421    0.0266    0.8836   -0.6906   -0.7560   -0.9621   -0.3212   -0.4990
   -0.4810   -0.7324   -1.2005    1.0095   -0.8362   -0.2506   -0.9675   -0.7560   -0.5634   -0.4167    0.3469
   -0.0613   -0.7324   -1.2005   -0.0421   -0.4115    0.5055   -1.0598    1.3510   -0.4712   -0.5402    0.0297
    1.6173    1.3640    0.8322    1.0095   -0.6371    1.4507   -0.5060   -0.7560   -0.4305    0.8774    2.2603

  Columns 12 through 22

   -0.7259    1.0134    0.4501   -0.0257    0.2908    0.6617    0.0636    1.3640    1.1070   -1.5232   -0.7606
    0.5677   -0.0037   -0.0428   -0.0275   -0.4750   -0.4925   -0.6030   -0.7324   -0.6463    0.0477   -0.0104
    0.6145   -0.4647    0.3392   -0.3444    0.2333    0.3469    0.2656   -0.7324    0.8498   -0.7468    0.6141
    0.1056    0.0259   -0.0346   -0.4527   -0.0796   -0.0961    0.0280   -0.7324    0.8498    0.0477    0.3067
    1.3908    1.6643   -1.0814    1.2469   -0.7702   -1.2269   -0.6837    1.3640    1.1070    1.4364   -0.8501

 Columns 23 through 33

  -0.8452   -0.8480   -1.0226   -0.0726   -0.8322   -0.8832   -0.5056   -0.4806   -0.4795   -0.6106   -0.0196
  -0.6331    0.4843    0.5678    0.7397   -0.8322   -0.0143   -0.0042    0.7770   -0.5556   -0.6106   -0.7502
   0.2041    0.6886    0.5678    0.4444    1.2005   -1.1828    0.9861    0.3428    1.1641    0.9161    0.7071
  -0.3541    0.7568   -0.9807    0.3762    1.2005    0.0706    0.4717   -0.5642    1.2739   -1.5969    0.5979
   2.0135   -0.7211   -1.0226   -0.5663   -0.8322    0.8546   -0.5613    1.2485   -0.4034   -0.6106   -0.3136

 Columns 34 through 44

   0.1279    0.5506    0.7064    0.6900    0.9188    0.0772   -0.3691    0.3839    0.3081    0.4460   -0.0180
  -0.6467   -0.0525   -0.0159    0.0533    0.0883    0.0386   -0.5472    0.0302   -0.0431   -0.0157   -0.0903
   0.0131   -0.8655   -0.2309   -0.9708   -0.7421   -0.5701   -0.1643    0.2265    0.7645    0.6512    0.4463
  -0.6467   -0.0347    0.0000    0.0691   -0.0044    0.0179   -0.4548   -0.2140    0.4004   -0.5635    0.1468
   0.0131   -0.6716    1.4806   -0.4964   -0.7421   -0.4360   -0.3251   -0.9696    0.2895    0.4973    0.2337

 Columns 45 through 55

  -0.4176    0.1772    0.4658    0.0024   -0.4768    0.3876    0.0651   -0.2638    0.0665   -0.7391         0
  -0.1510   -0.6101   -0.7019   -0.9048   -0.3883    0.4545    0.6782   -0.2638    0.7273   -0.0545    1.0000
  -0.6476    0.0705    0.1739    0.0993   -0.0468    0.6551    0.5604   -0.2638    0.4311   -0.5069         0
  -0.5143   -0.5507    0.6779   -0.2849    0.0924   -1.4178    0.5152    0.5099   -0.6125   -0.5779    1.0000
   0.7648   -0.7090   -1.1398   -0.6762   -0.5526    0.3208    0.2374   -0.2638    0.3324   -0.6052    1.0000
```

Рис. 3.2 – Вигляд даних вибірки на прикладі даних 5 клієнтів до стандартизації та після для нелінійної моделі

Дані в ході виконання програм також були розбиті на тренувальні і тестові у співвідношенні 0.7/0.3.

Далі опишемо основні функції, які використали в коді, за допомогою блок-схем. Для кожної моделі виведемо результати навчання та тестування алгоритмів.

3.2. Розв’язок задачі на основі лінійних моделей

3.2.1. Логістична функція втрат

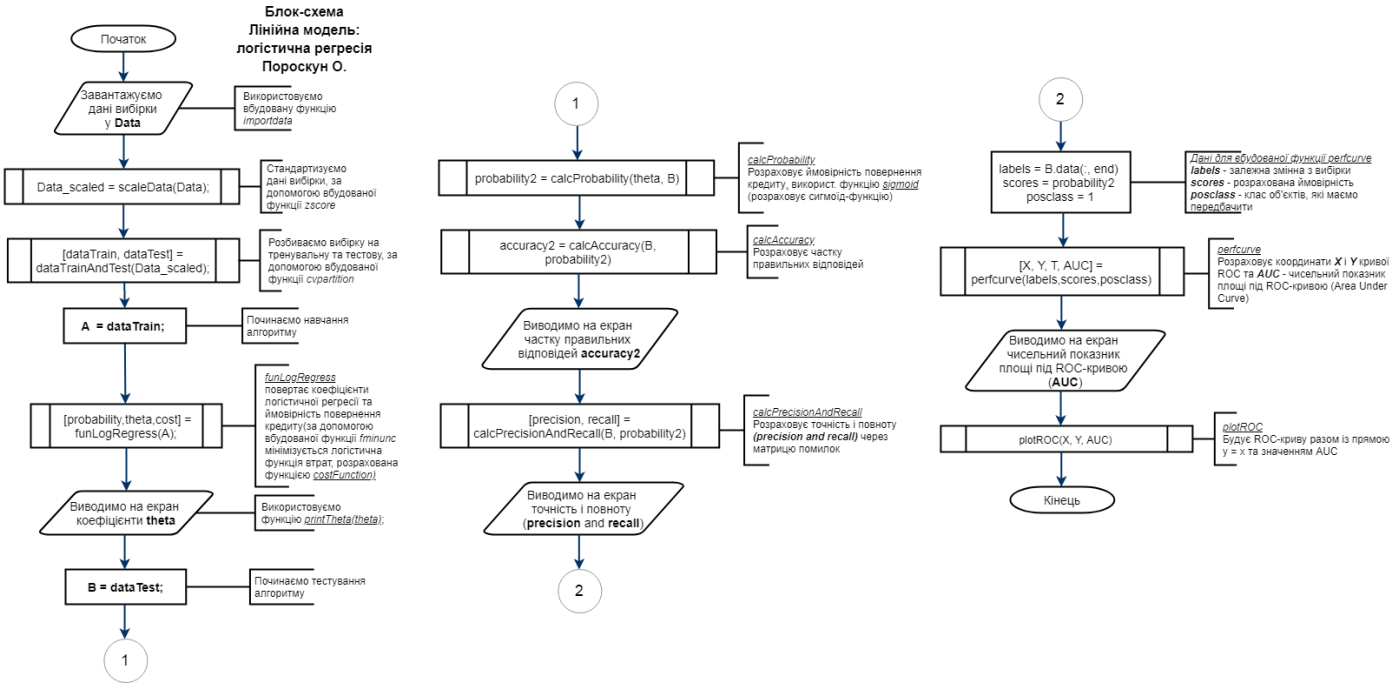


Рис. 3.3 – Блок-схема комп’ютерної моделі(1.1)

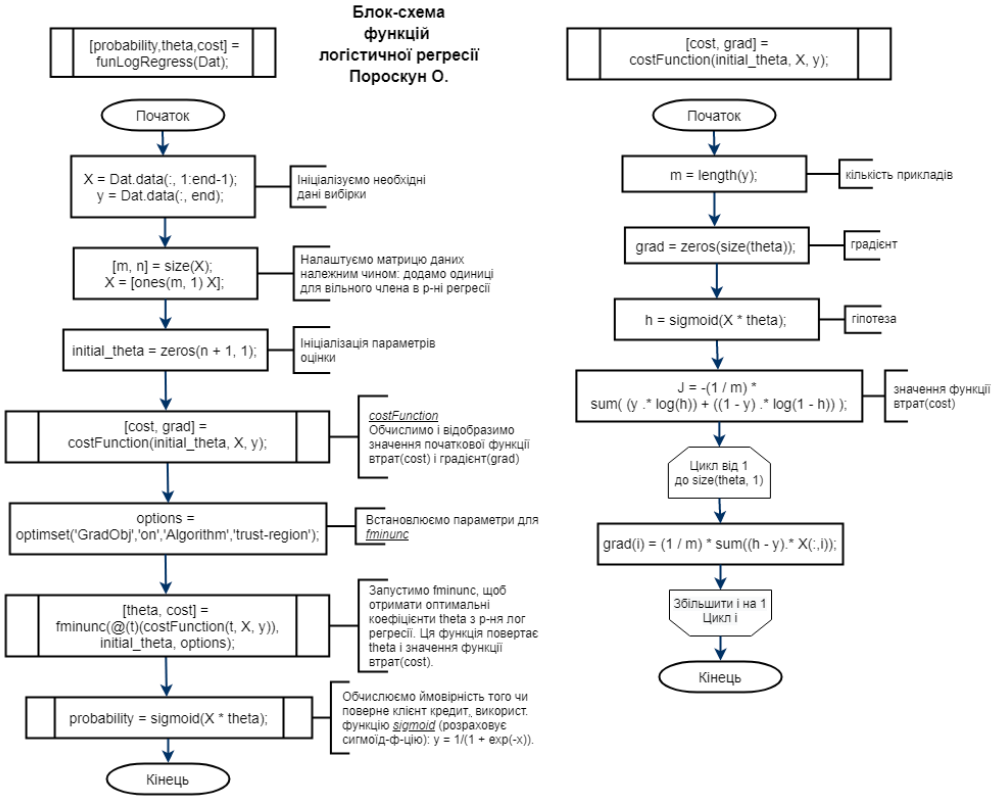


Рис. 3.4 – Блок-схема комп’ютерної моделі(1.2)

Результати виконання програми

```
РЕЗУЛЬТАТИ НАВЧАННЯ АЛГОРТИМУ

Навчальна вибірка(700 клієнтів).

Коефіцієнти логістичної регресії(theta):

  0.933968  0. константа
 -0.218793  1. вік
 -0.264913  2. стать
 -0.100326  3. шлюб
 -1.738594  4. утриманці
  5.118923  5. дохід
  0.083829  6. досвід роботи
  0.139060  7. термін проживання
  0.241566  8. нерухомість
 -1.384616  9. місячний платіж

ТЕСТУВАННЯ АЛГОРТИМУ

Тестова вибірка(299 клієнтів).

Оцінюємо якість побудованої лінійної моделі
(логістична регресія):

- accuracy (частка правильних відповідей): 0.903010

- precision(точність): 0.909774

- recall(повнота): 0.876812

- AUC(чисельний показник площі під ROC-кривою): 0.970114

Побудова графіку ROC-кривої...
```

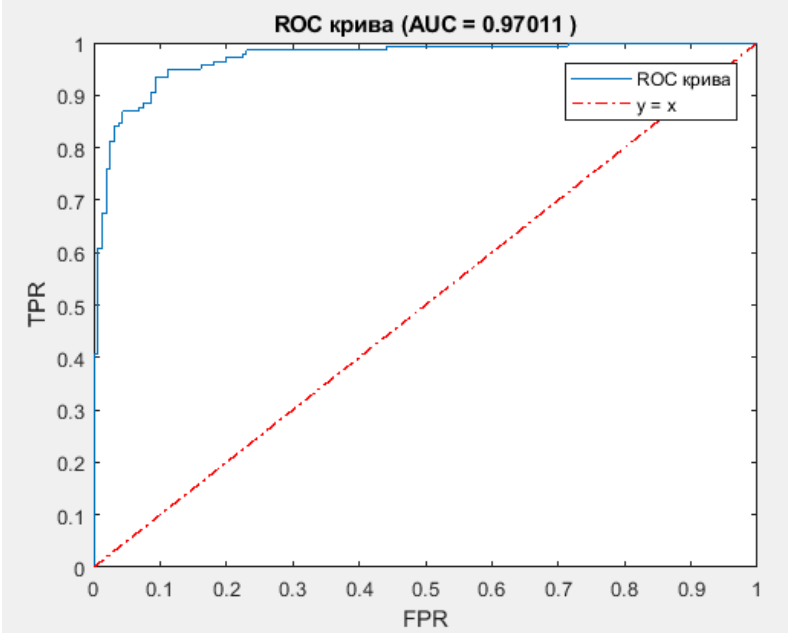


Рис. 3.5 – Результати роботи програми **linear_logistic.m**

3.2.2. Середньоквадратична функція втрат

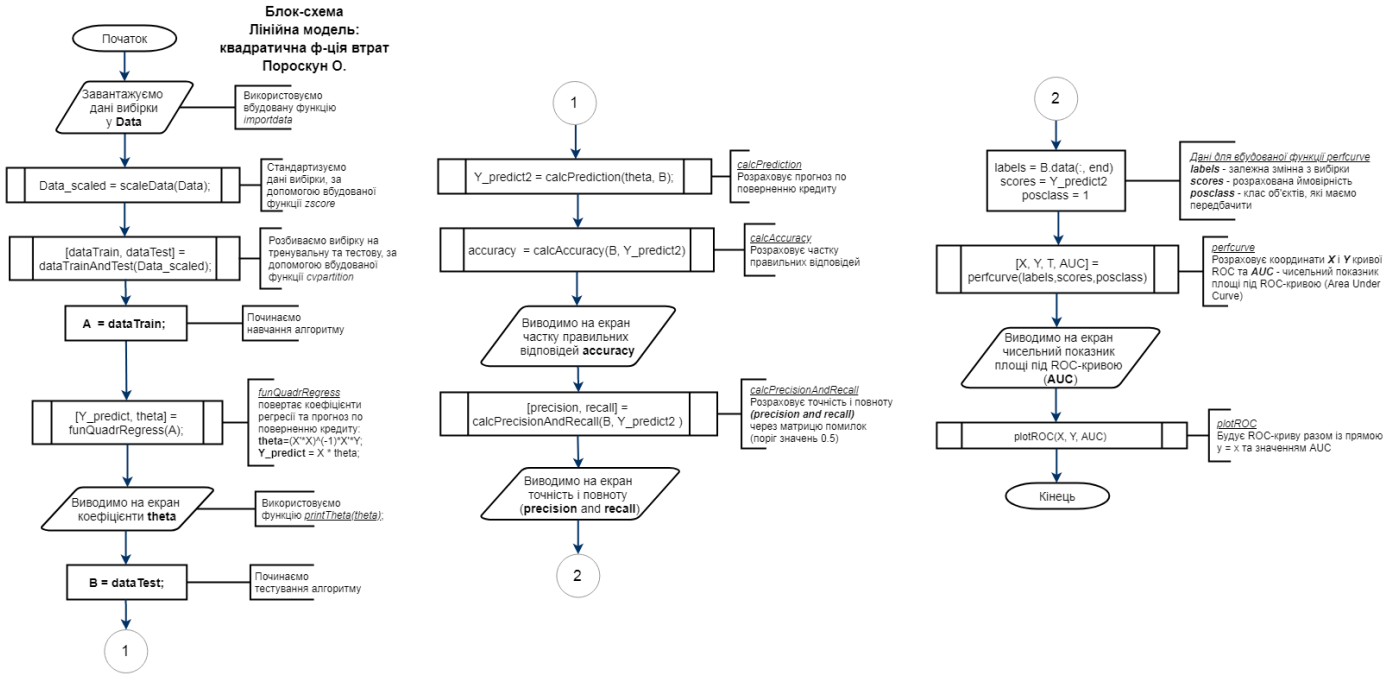
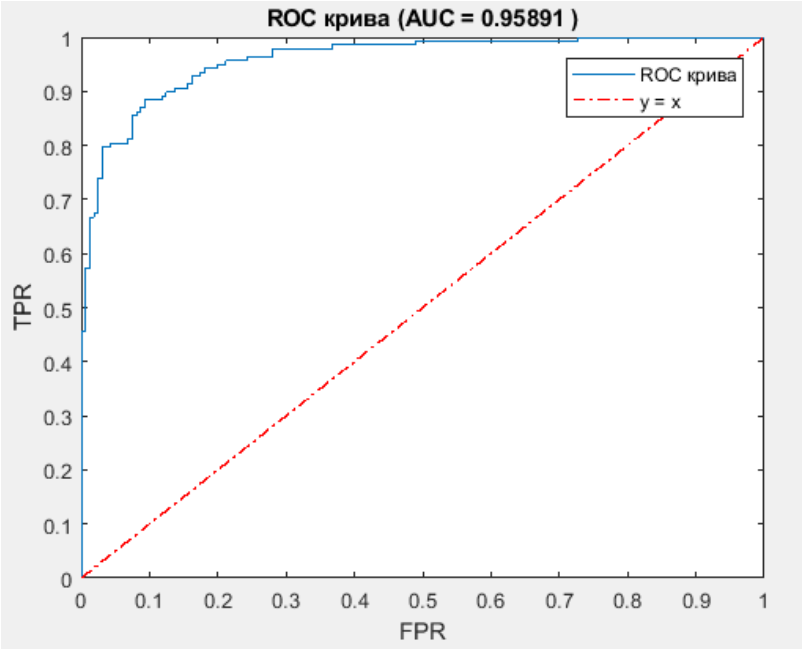


Рис. 3.6 – Блок-схема комп’ютерної моделі(2)

Результати виконання програми



```
РЕЗУЛЬТАТИ НАВЧАННЯ АЛГОРТИМУ

Навчальна вибірка(700 клієнтів).

Коефіцієнти регресії(theta):

  0.497551  0. константа
 -0.011140  1. вік
 -0.037421  2. стать
  0.005807  3. шлюб
 -0.150697  4. утриманці
  0.316541  5. дохід
  0.009382  6. досвід роботи
  0.003424  7. термін проживання
  0.029928  8. нерухомість
 -0.117874  9. місячний платіж

ТЕСТУВАННЯ АЛГОРТИМУ

Тестова вибірка(299 клієнтів).

Оцінюємо якість побудованої лінійної моделі
(квадратична функція втрат):

- accuracy (частка правильних відповідей): 0.889632

- precision(точність): 0.956522

- recall(повнота): 0.797101

- AUC(чисельний показник площі під ROC-кривою): 0.958907

Побудова графіку ROC-кривої...
```

Рис. 3.7 – Результати роботи програми **linear_quadratic.m**

3.3. Розв’язок задачі на основі нелінійних моделей

3.3.1. Логістична функція втрат

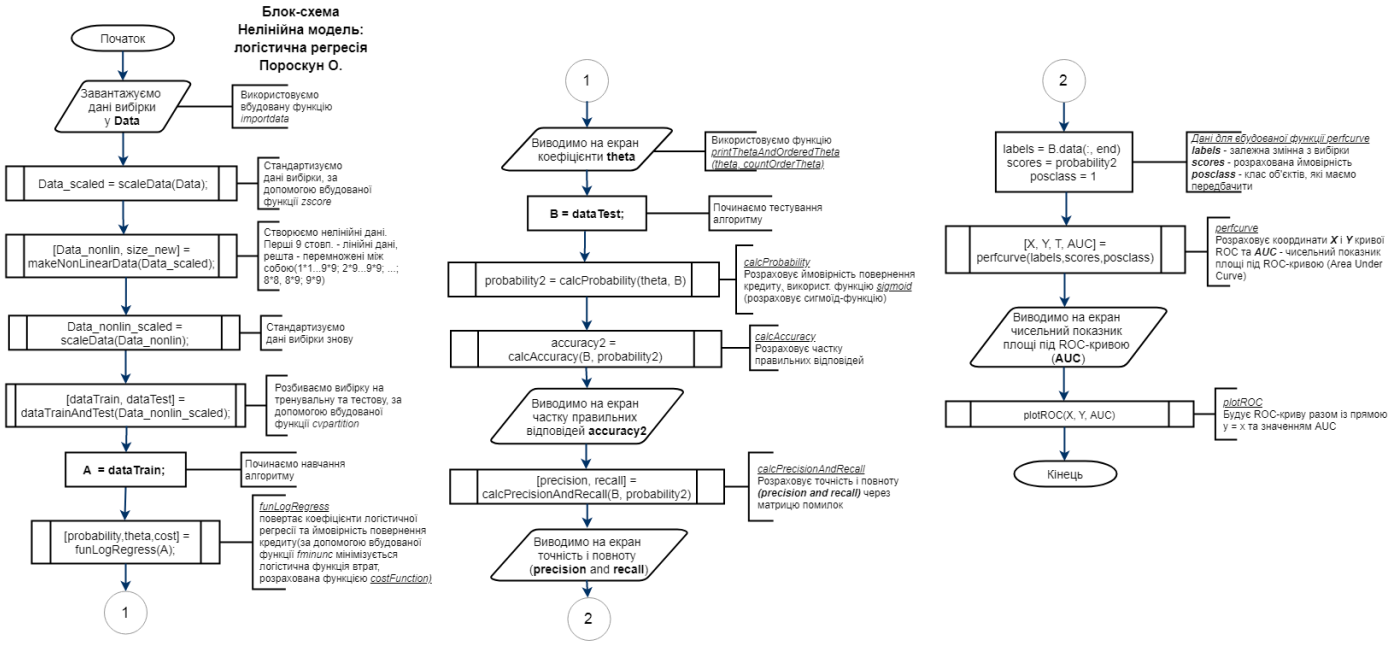


Рис. 3.8 – Блок-схема комп’ютерної моделі(3)

```
РЕЗУЛЬТАТИ НАВЧАННЯ АЛГОРТИМУ

Навчальна вибірка(700 клієнтів).

Коефіцієнти логістичної регресії(theta):

Коефіцієнт  №   Змінна

  0.621258  0   [0. константа]
 -0.444233  1   [1. вік]
 -0.264440  2   [2. стать]
 -0.103607  3   [3. шлюб]
 -2.480291  4   [4. утриманці]
  7.381914  5   [5. дохід]
  0.074431  6   [6. досвід роботи]
  0.186488  7   [7. термін проживання]
  0.785040  8   [8. нерухомість]
 -1.742463  9   [9. місячний платіж]
  0.812159  10  [1. вік x 1. вік]
 -0.076925  11  [1. вік x 2. стать]
  0.226570  12  [1. вік x 3. шлюб]
  0.146878  13  [1. вік x 4. утриманці]
 -0.102781  14  [1. вік x 5. дохід]
 -0.428284  15  [1. вік x 6. досвід роботи]
  0.272465  16  [1. вік x 7. термін проживання]
  0.111581  17  [1. вік x 8. нерухомість]
  0.261814  18  [1. вік x 9. місячний платіж]
 -0.264440  19  [2. стать x 2. стать]
 -0.071496  20  [2. стать x 3. шлюб]
 -0.197378  21  [2. стать x 4. утриманці]
 -0.202230  22  [2. стать x 5. дохід]
  0.304179  23  [2. стать x 6. досвід роботи]
 -0.231990  24  [2. стать x 7. термін проживання]
 -0.182667  25  [2. стать x 8. нерухомість]
```

```
 0.249833  26  [2. стать x 9. місячний платіж]
 0.103607  27  [3. шлюб x 3. шлюб]
 1.177749  28  [3. шлюб x 4. утриманці]
-1.520711  29  [3. шлюб x 5. дохід]
-0.172052  30  [3. шлюб x 6. досвід роботи]
-0.152042  31  [3. шлюб x 7. термін проживання]
-0.026116  32  [3. шлюб x 8. нерухомість]
 0.392616  33  [3. шлюб x 9. місячний платіж]
-0.710698  34  [4. утриманці x 4. утриманці]
-0.350494  35  [4. утриманці x 5. дохід]
-0.625575  36  [4. утриманці x 6. досвід роботи]
 0.261618  37  [4. утриманці x 7. термін проживання]
 0.367517  38  [4. утриманці x 8. нерухомість]
-0.316102  39  [4. утриманці x 9. місячний платіж]
-1.123144  40  [5. дохід x 5. дохід]
-0.105844  41  [5. дохід x 6. досвід роботи]
 0.898905  42  [5. дохід x 7. термін проживання]
-0.014583  43  [5. дохід x 8. нерухомість]
 1.177912  44  [5. дохід x 9. місячний платіж]
-0.461750  45  [6. досвід роботи x 6. досвід роботи]
-0.211434  46  [6. досвід роботи x 7. термін проживання]
-0.306124  47  [6. досвід роботи x 8. нерухомість]
-0.300927  48  [6. досвід роботи x 9. місячний платіж]
 0.792433  49  [7. термін проживання x 7. термін проживання]
-0.656963  50  [7. термін проживання x 8. нерухомість]
-0.174281  51  [7. термін проживання x 9. місячний платіж]
-0.599409  52  [8. нерухомість x 8. нерухомість]
 0.053469  53  [8. нерухомість x 9. місячний платіж]
 0.030759  54  [9. місячний платіж x 9. місячний платіж]

Коефіцієнти(theta) впорядковані за модулем(перші 5 з 55 шт):

Коефіцієнт  Змінна

  7.381914  [5. дохід]
 -2.480291  [4. утриманці]
 -1.742463  [9. місячний платіж]
 -1.520711  [3. шлюб x 5. дохід]
  1.177912  [5. дохід x 9. місячний платіж]

ТЕСТУВАННЯ АЛГОРТИМУ

Тестова вибірка(299 клієнтів).

Оцінюємо якість побудованої нелінійної моделі
(логістична регресія):

- accuracy (частка правильних відповідей): 0.842809

- precision(точність): 0.841060

- recall(повнота): 0.846667

- AUC(чисельний показник площі під ROC-кривою): 0.944072

Побудова графіку ROC-кривої...
```

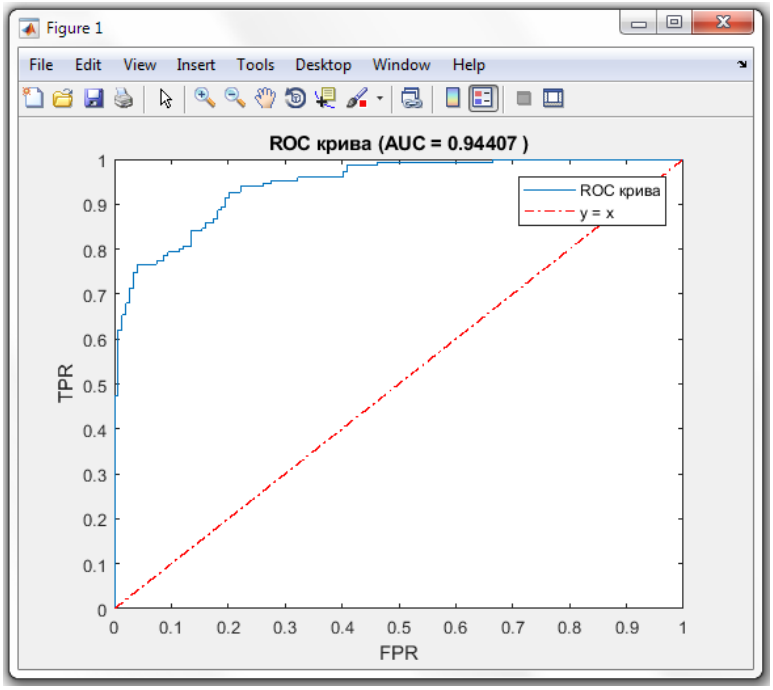


Рис. 3.9 – Результати роботи програми **non_linear_logistic.m**

3.4. Порівняння якості результатів, що отримані за допомогою різних моделей

	Міри якості моделі			
Назва моделі	accuracy (частка правильних відповідей)	precision (точність)	recall (повнота)	AUC(чисельний показник площі під ROC-кривою)
Лінійна з логістичною функцією втрат	0.903	0.9098	0.877	0.97
Лінійна з квадратичною функцією втрат	0.8896	0.9565	0.797	0.959
Нелінійна з логістичною функцією втрат	0.843	0.841	0.847	0.944

Табл. 1 – Оцінка якості побудованих регресійних моделей

ВИСНОВКИ

У рамках задачі кредитного скорингу проаналізовано якість регресійних моделей. Розглянувши значення мір якості побудованих моделей можна сказати про те, що відповідність прогнозованих та реальних результатів достатньо велика (90%).

Було розглянуто лінійні та нелінійні моделі з двома різними функціями втрат. З'ясовано, що лінійні моделі працюють краще (90%) за нелінійні (84%).

Визначено вплив факторних ознак на успішне повернення кредиту / дефолт про кредиту:

- *підтверджений сукупний наявний дохід* клієнта впливає позитивно на те, чи поверне він кредит чи ні: чим більший дохід, тим більша ймовірність того, що кредит буде повернуто;
- *кількість утриманців* впливає негативно: чим їх більше, тим скоріше за все клієнт не поверне кредит.
- *щомісячний платіж по кредиту* негативно впливає на повернення кредиту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 https://uk.education-wiki.com/9892350-what-is-regression-analysis

2 Купалова Г. І. Теорія економічного аналізу: навчальний посібник / Г. І. Купалова. — К.: Знання, 2008. — 639 с.

3 Zweig M.H., Campbell G. ROC Plots: A Fundamental Evaluation Tool in Clinical Medicine // Clinical Chemistry, Vol. 39, No. 4, 1993.

4 Fawcett T. ROC Graphs: Notes and Practical Considerations for Researchers // 2004 Kluwer Academic Publishers.

5 Davis J., Goadrich M. The Relationship Between Precision-Recall and ROC Curves // Proc. Of 23 International Conference on Machine Learning, Pittsburgh, PA, 2006.

6 Цыплаков А. А. Некоторые эконометрические методы. Метод максимального правдоподобия в эконометрии. Учебное пособие.

7 Mester, Loretta. (1997). What Is the Point of Credit Scoring?. Business Review. Sep/Oct. 3-16.

8 Безсмертний І. А. Штучний інтелект. СПб .: Изд-во СПбГУ ІТМО 2010.

9 Ширалієв, А. Е. Алгоритми машинного навчання для скорингових систем : магістерська дис. : 124 Системний аналіз / Ширалієв Анар Ельдар огли. - Київ, 2019. - 117 с. [Електронний ресурс]. - URL: https://ela.kpi.ua/handle/123456789/29842

10 Harrington P. Machine learning in action / P. Harrington. — Dublin: Manning Publications, 2012. — 382 p.

11 https://www.coursera.org/learn/machine-learning-with-python-ru

12 Теслюк В.М. Градієнтні методи розв’язання оптимізаційних задач: Ч.3. Конспект лекцій з курсу “Методи синтезу та оптимізації” для студентів базового напрямку “Комп’ютерні науки”. – Львів: Самвидав кафедри САП Національного університету “Львівська політехніка”, реєстр. номер №4947 від 27.05.2013. – 67 с. [Електронний ресурс]. - URL: https://usnd.to/CzUt

13 https://www.coursera.org/learn/machine-learning

14 https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

15 Попко, А. В. Методи виявлення образ в коротких текстах : магістерська дис. : 151 Автоматизація та комп'ютерно-інтегровані технології / Попко Андрій Валентинович. – Київ, 2018. – 133 с. https://ela.kpi.ua/handle/123456789/23001

16 Воронцов К.В. Математические методы обучения по прецедентам. – 2007. – 141 с. [Електронний ресурс]. - URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf

17 Бессмертный, И. А. Интеллектуальные системы : учебник и практикум для академического бакалавриата / И. А. Бессмертный, А. Б. Нугуманова, А. В. Платонов. — Москва : Издательство Юрайт, 2019. — 243 с.

18 https://youtu.be/zMqhnpWgSS8

19 Інтелектуальний аналіз даних: методичні вказівки до виконання комп'ютерних практикумів з навчальної дисципліни «Інтелектуальний аналіз даних». Частина-1. «Кореляційний та регресійний аналіз медичних даних». / Уклад.: д.б.н., с.н.с. Є. А. Настенко, к.т.н. В. С. Якимчук, к.т.н. О. К. Носовець. – К.: НТУУ «КПІ ім. І. Сікорського», 2017. – 51 с.

20 Руденко В. М. (2012). Математична статистика. Київ: Центр учбової літератури. с. 158

21 https://stud.com.ua/139981/informatika/perehid_neliniynoyi_regresiyi#977

ДОДАТКИ

ДОДАТОК А

Лістинг програми **linear_logistic.m**

```
% Бакалаврська робота Пороскун О. ПМ-81
% (лінійна модель, логістична регресія)

close all; clear all; clc

% 1. Завантажуємо дані з файлу
%filename = input('Введіть назву файлу: ', 's');
filename = 'bank_new.txt'; delimiterIn = ' ';
Data = importdata(filename);

% number = 5; % кількість даних для друку  % printData(Data, number);

% 2. Стандартизуємо дані
Data_scaled = scaleData(Data);

% 3. Розбиваємо вибірку на тренувальну та тестову (трен: 70%, тест: 30%)
[dataTrain, dataTest] = dataTrainAndTest(Data_scaled);

% 4.1 Навчаємо алгоритм на трен. вибірці і знаходимо коефцієнти
%    логістичної регресії(theta)
A = dataTrain;

% Ініціалізація матриць прогнозів/ймовірності, що клієнт поверне
кредит/(probability),
% частки правильних відповідей(accuracy), коефіцієнтів (theta),
% значень функції втрат (cost)
probability = zeros(length(A.data), 1); cost = 0;
theta = zeros(size(A.data, 2), 1);

% Обчислюємо ініціалізовані раніше величини
[probability,theta,cost] = funLogRegress(A);

% Друкуємо результати навчання алгортиму
fprintf('РЕЗУЛЬТАТИ НАВЧАННЯ АЛГОРТИМУ\n\n');
fprintf('Навчальна вибірка(%d клієнтів).\n\n', length(A.data));
str = sprintf(['Коефіцієнти логістичної регресії(theta):\n']);
disp(str); clear str;
printTheta(theta);

% 4.2 Тестуємо навчений алгоритм, використовуючи отримані коєфіцієнти(theta)
B = dataTest;

probability2 = zeros(length(B.data), 1); accuracy = 0;

% Обчислюємо ймовірність того чи поверне клієнт кредит
probability2 = calcProbability(theta, B);

% Друкуємо результати тестування алгортиму
fprintf('\nТЕСТУВАННЯ АЛГОРТИМУ\n\n');
fprintf('Тестова вибірка(%d клієнтів).\n\n', length(B.data));
fprintf('Оцінюємо якість побудованої лінійної моделі\n');
fprintf('(логістична регресія):\n\n');
```

```
% Accuracy (частка правильних відповідей)
accuracy = calcAccuracy(B, probability2);
fprintf('- accuracy (частка правильних відповідей): %f\n\n', accuracy);

% Матриця похибок. Точність і повнота
[precision, recall] = calcPrecisionAndRecall(B, probability2);
fprintf('- precision(точність): %f\n\n', precision);
fprintf('- recall(повнота): %f\n\n', recall);

% ROC-крива та AUC
labels = B.data(:, end); % залежна змінна з вибірки
scores = probability2;   % розрахована ймовірність
posclass = 1;            % клас об'єктів, які маємо передбачити

% Розраховуємо за допомогою функції координати X і Y кривої ROC та AUC -
% чисельний показник площі під ROC-кривою (Area Under Curve)
[X,Y,T,AUC] = perfcurve(labels,scores,posclass);
fprintf('- AUC(чисельний показник площі під ROC-кривою): %f\n\n', AUC);

% Будуємо ROC-криву разом із прямою y = x.
plotROC(X, Y, AUC);


% disp('------------------ Functions (функції) ------------------');

% disp('--- Друк даних вибірки ---');
function printData(Dat, number)
    disp('Вхідні дані - вибірка анкетних даних позичальників.');
    fprintf('Перші %d позичальників мають такі анкетні дані.\nНазви
стовпців:\n', number);
    for i = 1:size(Dat.data, 2)
        fprintf('\t%d\t%s ', i, Dat.colheaders{1, i});
        if i == (size(Dat.data, 2) / 2 + 1)
            disp(' ');
        end
    end
    disp(' '); disp('Дані позичальників:'); disp(' ');
    disp(num2str(Dat.data(1:number, :)));disp(' '); % дані від 1 до number
клієнта
end

% % disp('--- Стандартизація даних ---');
function Data_scaled = scaleData(Dat)
    % scaleData(Data) проводить стандартизацію даних незал. змінних
    % z = zscore(X) - стандартизує дані на основі середнього значення та
    %     стандартного відхилення вихідних даних X (z=(x-mean(X))./std(X))
    Data_scaled = Dat;
    size_dat = size(Data_scaled.data);
    Data_scaled.data(:, 1:end-1) = 0;
    Dat_mean = zeros(1, size_dat(2)); Dat_std = zeros(1, size_dat(2));
    for i = 1:size_dat(2)-1
        [Data_scaled.data(:,i),Dat_mean(i),Dat_std(i)] = ...
           zscore(Dat.data(:, i));
    end
end

% disp('--- Розбиття даних вибірки ---');
function [dataTrain, dataTest] = dataTrainAndTest(Dat)
    % dataTrainAndTest(Dat) розбиває вибірку на тренувальну та тестову
    % cvpartition(size,'HoldOut',0.3) визначає випадковий розділ набору
    %                                 даних у заданому співвідношенні
```

```
    dataTrain = Dat; dataTest  = Dat;
    % Перехресна варіація (трен: 70%, тест: 30%)
    cv = cvpartition(size(Dat.data,1),'HoldOut',0.3);
    idx = cv.test;
    % Відокремлення тренування та тестування
    dataTrain.data = Dat.data(~idx,:);
    dataTest.data  = Dat.data(idx,:);
end

% disp('--- Знаходження даних для лог. регресії ---');
function [probability, theta, cost] = funLogRegress(Dat)
    X = Dat.data(:, 1:end-1);
    y = Dat.data(:, end);

    % Налаштуємо матрицю даних належним чином: додамо одиниці для вільного члена
в р-ні регресії
    [m, n] = size(X);
    X = [ones(m, 1) X];

    % Ініціалізація параметрів оцінки
    initial_theta = zeros(n + 1, 1);

    % Обчислимо і відобразимо значення початкової функції втрат(cost) і
градієнт(grad)
    [cost, grad] = costFunction(initial_theta, X, y);

    % Встановлюємо параметри для fminunc
    options = optimset('GradObj','on','Algorithm','trust-region');

    % Запустимо fminunc, щоб отримати оптимальні коефіцієнти theta з р-ня лог
регресії
    % z = X * theta = x(1)*theta(1) + x(2)*theta(2) + ... + x(10)*theta(10)
    % Ця функція повертає theta і значення функції втрат(cost).
    [theta, cost] = ...
    fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);

    %fprintf('Кількість об`єктів у вибірці: %d\n\n', length(A.data));
    %fprintf('Функцію втрат(помилки) в theta знайдено через fminunc: %f\n\n',
cost);

    %disp('Коефіцієнти для лінійної частини логістичної регресії:');
    %fprintf('theta: \n'); fprintf(' %f \n', theta);disp(' ');

    % Обчислюємо ймовірність того чи поверне клієнт кредит
    probability = sigmoid(X * theta);

    % Обчислюємо точність на нашому наборі даних
    %prediction = predict(theta, X); % або round(probability)
end

% disp('--- Розрахування функції втрат та градієнта ---');
function [J, grad] = costFunction(theta, X, y)
    %  costFunction(theta, X, y) обчислює: J - значення функції втрат(cost)
    %  при використанні theta як параметру для логістичної регресії та
    %  grad - градієнта втрат, де X = [ones() X];
    %  theta з найменшим J є найкращою theta

    % кількість прикладів
    m = length(y);

    % градієнт
    grad = zeros(size(theta));
```

```
    % гіпотеза
    h = sigmoid(X * theta);

    J = -(1 / m) * sum( (y .* log(h)) + ((1 - y) .* log(1 - h)) );

    for i = 1 : size(theta, 1)
        grad(i) = (1 / m) * sum((h - y).* X(:,i));
    end

end

% disp('--------- Друк коефіцієнтів регресії ---------');
function printTheta(theta)
    % printTheta(theta) друкує коефіцієнти theta
    str_koef(1:10) = ...
       ["0. константа"; "1. вік"; "2. стать"; "3. шлюб"; "4. утриманці";
        "5. дохід"; "6. досвід роботи"; "7. термін проживання";
        "8. нерухомість"; "9. місячний платіж"];
    for j = 1:10
        fprintf(' ');
        if theta(j) > 0
            fprintf(' ');
        end
        fprintf('%f\t%s\n', theta(j), str_koef(j));
    end
end

% disp('--- Розрахування сігмоїдної функції ---');
function g = sigmoid(z)
    %  Розраховуємо сигмовидну функцію
    %  g = sigmoid(z) він обчислює сигмоїду z.
    g = 1 ./ (1 + exp(-z));
end

% disp('--- Передбачення чи повернуть кредит -> 0 або 1 ---');
function p = predict(theta, X)
    % Передбачаємо, чи буде мітка 0 чи 1, використовуючи навчені логістики
    % параметри(коефіцієнти) регресії theta
    p = round(sigmoid(X * theta));
    % p = predict(theta, X) обчислює передбачення для X за допомогою
    % порога на рівні 0,5 (тобто, якщо sigmoid(theta'*x) >= 0.5, передбачити 1)
end

% disp('--- Розрахування ймовірності ---');
function probability = calcProbability(theta, Dat)
    % Ініціалізація матриці даних незалежних змінних
    X = Dat.data(:, 1:end-1);

    % Налаштуємо матрицю даних належним чином:
    % додамо одиниці для вільного члена в р-ні регресії
    [m, n] = size(X);
    X = [ones(m, 1) X];

    % Обчислюємо ймовірність того чи поверне клієнт кредит
    probability = sigmoid(X * theta);
end

% disp('--------- Розрахування accuracy ---------');
function accuracy = calcAccuracy(Dat, probability)
    % Розрахування accuracy (частки правильних відповідей )
    % 2 способи обчислення:
```

```
    % accuracy_1 = (TP+TN)/(TP+TN+FP+FN)(через матрицю помилок)
    % accuracy_2 = sum(OrigValues == PredValues) / numel(PredValues)
                % Функція numel(A) повертає кількість елементів у масиві A.
    OrigValues = Dat.data(:, end);
    PredValues = round(probability);
    accuracy = sum(OrigValues == PredValues) / numel(PredValues);
end

% disp('--------- Розрахування precision та recall---------');
function [precision, recall] = calcPrecisionAndRecall(Dat, probability)
    % precision (точність)
    % recall (повнота)

    % Розрахування confusion matrix (матриці помилок)
    %           y = 1                   y = 0
    % y' = 1    True Positive (TP)      False Positive (FP)
    % y' = 0    False Negative (FN)     True Negative (TN)

    TP = 0; FN = 0; FP = 0; TN = 0;
    for i = 1:length(Dat.data)
        if Dat.data(i, size(Dat.data, 2)) == 1
            if probability(i) >= 0.5
                TP = TP + 1;
            else
                FN = FN + 1;
            end
        else
            if probability(i) >= 0.5
                FP = FP + 1;
            else
                TN = TN + 1;
            end
        end
    end
    %disp('confusion matrix (матриці помилок):');disp(' ');
    %fprintf('TP: %d\t FP: %d\nFN: %d\t TN: %d\n',TP, FP, FN, TN);
    %fprintf('Їх сума(має = %d): %d\n\n', length(A.data), TP+FP+FN+TN);

    precision = TP / (TP + FP);
    recall = TP / (TP + FN);
end

% disp('------------- Побудова графіку ROC-кривої -------------');
function plotROC(X, Y, AUC)
    disp('Побудова графіку ROC-кривої...');
    % Будуємо ROC-криву разом із прямою y = x.
    figure;
    plot(X, Y);   hold on;
    xx = [0 1]; yy = [0 1];
    plot(xx, yy, 'r-.', 'linewidth', 1);
    xlabel('FPR'); % False Positive Rate
    ylabel('TPR'); % True Positive Rate
    title(['ROC крива (AUC = ' num2str(AUC) ' )']);
    legend('ROC крива', 'y = x');
end
```

ДОДАТОК Б

Лістинг програми **linear_quadratic.m**

```
% Бакалаврська робота Пороскун О. ПМ-81
% (лінійна модель, квадратична ф-ція втрат)

close all; clear all; clc

% 1. Завантажуємо дані з файлу
%filename = input('Введіть назву файлу: ', 's');
filename = 'bank_new.txt'; delimiterIn = ' ';
Data = importdata(filename);

% number = 5; % кількість даних для друку  % printData(Data, number);

% 2. Стандартизуємо дані
Data_scaled = scaleData(Data);

% 3. Розбиваємо вибірку на тренувальну та тестову (трен: 70%, тест: 30%)
[dataTrain, dataTest] = dataTrainAndTest(Data_scaled);

% 4.1 Навчаємо алгоритм на трен. вибірці і знаходимо коефцієнти регресії(theta)
A = dataTrain;

% Ініціалізація матриць прогнозів, що клієнт поверне кредит(Y_predict),
% коефіцієнтів (theta)
Y_predict = zeros(length(A.data), 1);
theta = zeros(size(A.data, 2), 1);

% Обчислюємо ініціалізовані раніше величини
[Y_predict, theta] = funQuadrRegress(A);

% Друкуємо результати навчання алгортиму
fprintf('РЕЗУЛЬТАТИ НАВЧАННЯ АЛГОРТИМУ\n\n');
fprintf('Навчальна вибірка(%d клієнтів).\n\n', length(A.data));
str = sprintf('Коефіцієнти регресії(theta):\n');
disp(str); clear str;
printTheta(theta);


% 4.2 Тестуємо навчений алгоритм, використовуючи отримані коєфіцієнти(theta)
B = dataTest;

Y_predict2 = zeros(length(B.data), 1); %accuracy = 0;

% Обчислюємо ймовірність того чи поверне клієнт кредит
Y_predict2 = calcPrediction(theta, B);

% Друкуємо результати тестування алгортиму
fprintf('\nТЕСТУВАННЯ АЛГОРТИМУ\n\n');
fprintf('Тестова вибірка(%d клієнтів).\n\n', length(B.data));
fprintf('Оцінюємо якість побудованої лінійної моделі\n');
fprintf('(квадратична функція втрат):\n\n');

% Accuracy (частка правильних відповідей)
accuracy = calcAccuracy(B, Y_predict2);
fprintf('- accuracy (частка правильних відповідей): %f\n\n', accuracy);

% Матриця похибок. Точність і повнота
[precision, recall] = calcPrecisionAndRecall(B, Y_predict2);
fprintf('- precision(точність): %f\n\n', precision);
```

```
fprintf('- recall(повнота): %f\n\n', recall);

% ROC-крива та AUC
labels = B.data(:, end); % залежна змінна з вибірки
scores = Y_predict2;     % розраховане передбачення
posclass = 1;            % клас об'єктів, які маємо передбачити

% Розраховуємо за допомогою функції координати X і Y кривої ROC та AUC -
% чисельний показник площі під ROC-кривою (Area Under Curve)
[X,Y,T,AUC] = perfcurve(labels,scores,posclass);
fprintf('- AUC(чисельний показник площі під ROC-кривою): %f\n\n', AUC);

% Будуємо ROC-криву разом із прямою y = x.
plotROC(X, Y, AUC);


% disp('------------------ Functions (функції) ------------------');

% disp('--- Друк даних вибірки ---');
function printData(Dat, number)
    disp('Вхідні дані - вибірка анкетних даних позичальників.');
    fprintf('Перші %d позичальників мають такі анкетні дані.\nНазви
стовпців:\n', number);
    for i = 1:size(Dat.data, 2)
        fprintf('\t%d\t%s ', i, Dat.colheaders{1, i});
        if i == (size(Dat.data, 2) / 2 + 1)
            disp(' ');
        end
    end
    disp(' '); disp('Дані позичальників:'); disp(' ');
    disp(num2str(Dat.data(1:number, :)));disp(' '); % дані від 1 до number
клієнта
end

% disp('--- Стандартизація даних ---');
function Data_scaled = scaleData(Dat)
    % scaleData(Data) проводить стандартизацію даних незал. змінних
    % z = zscore(X) - стандартизує дані на основі середнього значення та
    %     стандартного відхилення вихідних даних X (z=(x-mean(X))./std(X))
    Data_scaled = Dat;
    size_dat = size(Data_scaled.data);
    Data_scaled.data(:, 1:end-1) = 0;
    Dat_mean = zeros(1, size_dat(2)); Dat_std = zeros(1, size_dat(2));
    for i = 1:size_dat(2)-1
        [Data_scaled.data(:,i),Dat_mean(i),Dat_std(i)] = ...
           zscore(Dat.data(:, i));
    end
end

% disp('--- Розбиття даних вибірки ---');
function [dataTrain, dataTest] = dataTrainAndTest(Dat)
    % dataTrainAndTest(Dat) розбиває вибірку на тренувальну та тестову
    % cvpartition(size,'HoldOut',0.3) визначає випадковий розділ набору
    %                                 даних у заданому співвідношенні
    dataTrain = Dat; dataTest  = Dat;
    % Перехресна варіація (трен: 70%, тест: 30%)
    cv = cvpartition(size(Dat.data,1),'HoldOut',0.3);
    idx = cv.test;
    % Відокремлення тренування та тестування
    dataTrain.data = Dat.data(~idx,:);
    dataTest.data  = Dat.data(idx,:);
end
```

```
% disp('--- Знаходження даних для лін. регресії ---');
function [Y_predict, theta] = funQuadrRegress(Dat)
    X = Dat.data(:, 1:end-1);
    y = Dat.data(:, end);

    % Налаштуємо матрицю даних належним чином:
    % додамо одиниці для вільного члена в р-ні регресії
    [m, n] = size(X);
    X = [ones(m, 1) X];

    inv_X = inv(X'*X);
    theta = inv_X*X'*y; % за формулою theta=(X'*X)^(-1)*X'*Y;

    % Обчислюємо прогнозовані значення
    Y_predict = X * theta;
end

% disp('--------- Друк коефіцієнтів регресії ---------');
function printTheta(theta)
    % printTheta(theta) друкує коефіцієнти theta
    str_koef(1:10) = ...
       ["0. константа"; "1. вік"; "2. стать"; "3. шлюб"; "4. утриманці";
        "5. дохід"; "6. досвід роботи"; "7. термін проживання";
        "8. нерухомість"; "9. місячний платіж"];
    for j = 1:10
        fprintf(' ');
        if theta(j) > 0
            fprintf(' ');
        end
        fprintf('%f\t%s\n', theta(j), str_koef(j));
    end
end

% disp('--- Розрахування передбачення ---');
function Y_predict = calcPrediction(theta, Dat)
    % Ініціалізація матриці даних незалежних змінних
    X = Dat.data(:, 1:end-1);

    % Налаштуємо матрицю даних належним чином:
    % додамо одиниці для вільного члена в р-ні регресії
    [m, n] = size(X);
    X = [ones(m, 1) X];

    % Обчислюємо передбачення того чи поверне клієнт кредит
    Y_predict = X * theta;
end

% disp('--- Передбачення чи повернуть кредит -> 0 або 1 ---');
function bp = binary_prediction(Y_predict)
    % binary_prediction() обчислює бінарне передбачення за допомогою
    % threshold - порога на рівні 0,5 (якщо Y_predict >= 0.5, передбачити 1)
    bp = Y_predict;
    threshold = 0.5; % поріг для класифікації передбачених значень
    for i = 1:length(Y_predict)
        if Y_predict(i) >= threshold
            bp(i) = 1;
        else
            bp(i) = 0;
        end
    end
end
```

```
% disp('--------- Розрахування accuracy ---------');
function accuracy = calcAccuracy(Dat, Y_predict)
    % Розрахування accuracy (частки правильних відповідей )
    % 2 способи обчислення:
    % accuracy_1 = (TP+TN)/(TP+TN+FP+FN)(через матрицю помилок)
    % accuracy_2 = sum(OrigValues == PredValues) / numel(PredValues)
                % Функція numel(A) повертає кількість елементів у масиві A.
    OrigValues = Dat.data(:, end);
    PredValues = binary_prediction(Y_predict);

    accuracy = sum(OrigValues == PredValues) / numel(PredValues);
end

% disp('--------- Розрахування precision та recall---------');
function [precision, recall] = calcPrecisionAndRecall(Dat, Y_predict)
    % precision (точність)
    % recall (повнота)

    % Розрахування confusion matrix (матриці помилок)
    %           y = 1                 y = 0
    % y' = 1    True Positive (TP)    False Positive (FP)
    % y' = 0    False Negative(FN)    True Negative  (TN)

    % змінюємо значення відповідно до порога (0.5)
    Y_bin_predict = binary_prediction(Y_predict);

    TP = 0; FN = 0; FP = 0; TN = 0;
    for i = 1:length(Dat.data)
        if Dat.data(i, size(Dat.data, 2)) == 1
            if Y_bin_predict(i) == 1
                TP = TP + 1;
            else
                FN = FN + 1;
            end
        else
            if Y_bin_predict(i) == 1
                FP = FP + 1;
            else
                TN = TN + 1;
            end
        end
    end
    % disp('confusion matrix (матриці помилок):');disp(' ');
    %fprintf('TP: %d\t FP: %d\nFN: %d\t TN: %d\n',TP, FP, FN, TN);
    %fprintf('Їх сума(має = %d): %d\n\n', length(A.data), TP+FP+FN+TN);
    precision = TP / (TP + FP);
    recall = TP / (TP + FN);
end

% disp('------------- Побудова графіку ROC-кривої -------------');
function plotROC(X, Y, AUC)
    disp('Побудова графіку ROC-кривої...');
    % Будуємо ROC-криву разом із прямою y = x.
    figure;
    plot(X, Y);   hold on;
    xx = [0 1]; yy = [0 1];
    plot(xx, yy, 'r-.', 'linewidth', 1);
    xlabel('FPR'); % False Positive Rate
    ylabel('TPR'); % True Positive Rate
    title(['ROC крива (AUC = ' num2str(AUC) ' )']);
    legend('ROC крива', 'y = x');
end
```

ДОДАТОК В

Лістинг програми **non_linear_logistic.m**

```
% Бакалаврська робота Пороскун О. ПМ-81
% (нелінійна модель, логістична регресія)

close all; clear all; clc

% 1. Завантажуємо дані з файлу
%filename = input('Введіть назву файлу: ', 's');
filename = 'bank_new.txt'; delimiterIn = ' ';
Data = importdata(filename);

% number = 5; % printData(Data, number, "lin"); % друк лінійних даних

% 2. Стандартизуємо дані
Data_scaled = scaleData(Data);

% 3. Створюємо нелінійні дані
[Data_nonlin, size_new] = makeNonLinearData(Data_scaled);

% 4. Стандартизуємо дані знову
Data_nonlin_scaled = scaleData(Data_nonlin);
% printData(Data_nonlin_scaled, number, "nolin"); % друк нелінійних даних

% 5. Розбиваємо вибірку на тренувальну та тестову (трен: 70%, тест: 30%)
[dataTrain, dataTest] = dataTrainAndTest(Data_nonlin_scaled);


% 6.1. Навчаємо алгоритм на трен. вибірці і знаходимо коефіцієнти
%      логістичної регресії(theta)
A = dataTrain;

% Ініціалізація матриці ймовірності, що клієнт поверне кредит/(probability),
% частки правильних відповідей(accuracy), коефіцієнтів (theta),
% значень функції втрат (cost)
probability = zeros(length(A.data), 1); cost = 0;
theta = zeros(size(A.data, 2), 1);

% Обчислюємо ініціалізовані раніше величини
[probability,theta,cost] = funLogRegress(A);

% Друкуємо результати навчання алгортиму
fprintf('РЕЗУЛЬТАТИ НАВЧАННЯ АЛГОРТИМУ\n\n');
fprintf('Навчальна вибірка(%d клієнтів).\n\n', length(A.data));
str = sprintf('Коефіцієнти логістичної регресії(theta):\n');
disp(str); clear str;
countOrderTheta = 5; % к-сть впорядкованих коеф. theta для друку
printThetaAndOrderedTheta(theta, countOrderTheta);


% 6.2 Тестуємо навчений алгоритм, використовуючи отримані коєфіцієнти(theta)
B = dataTest;

% Ініціалізація
probability2 = zeros(length(B.data), 1); accuracy = 0;

% Обчислюємо ймовірність того чи поверне клієнт кредит
probability2 = calcProbability(theta, B);

% Друкуємо результати тестування алгортиму
fprintf('\nТЕСТУВАННЯ АЛГОРТИМУ\n\n');
fprintf('Тестова вибірка(%d клієнтів).\n\n', length(B.data));
```

```
fprintf('Оцінюємо якість побудованої нелінійної моделі\n');
fprintf('(логістична регресія):\n\n');

% Accuracy (частка правильних відповідей)
accuracy = calcAccuracy(B, probability2);
fprintf('- accuracy (частка правильних відповідей): %f\n\n', accuracy);

% Матриця похибок. Точність і повнота
[precision, recall] = calcPrecisionAndRecall(B, probability2);
fprintf('- precision(точність): %f\n\n', precision);
fprintf('- recall(повнота): %f\n\n', recall);

% ROC-крива та AUC
labels = B.data(:, end); % залежна змінна з вибірки
scores = probability2;   % розрахована ймовірність
posclass = 1;            % клас об'єктів, які маємо передбачити

% Розраховуємо за допомогою функції координати X і Y кривої ROC та AUC -
% чисельний показник площі під ROC-кривою (Area Under Curve)
[X,Y,T,AUC] = perfcurve(labels,scores,posclass);
fprintf('- AUC(чисельний показник площі під ROC-кривою): %f\n\n', AUC);

% Будуємо ROC-криву разом із прямою y = x.
plotROC(X, Y, AUC);


% disp('------------------ Functions (функції) ------------------');

% disp('--- Друк даних вибірки ---');
function printData(Dat, number, linOrNot)
    k = linOrNot;
    disp('Вхідні дані - вибірка анкетних даних позичальників.');
    fprintf('Перші %d позичальників мають такі анкетні дані.\nНазви
стовпців:\n', number);
    for i = 1:9
        fprintf('\t%d\t%s ', i, Dat.colheaders{1, i});
        if i == 6
            disp(' ');
        end
    end
    n = size(Dat.data, 2);
    if k == 'lin'
        fprintf('\t%d\t%s \n', n, Dat.colheaders{1, 10});
        fprintf('\nДані позичальників:\n\n');
        disp(num2str(Dat.data(1:number, :))); % дані від 1 до number клієнта
    end
    if k == 'nolin'
        fprintf('\nДалі йдуть стовпчики перемножені один на одного:\n');
        fprintf('\t10 1x1,  11 1x2, ...,  18 1x9 (9 шт),  19 2x2, ...,  26 2x9
(8 шт), ...,  54 9x9(1 шт)');
        fprintf('\n\t%d\t%s \n', n, Dat.colheaders{1, 10});
        disp(' '); disp('Дані позичальників:');
        Dat.data(1:number, :) % дані від 1 до number клієнта
    end
    disp(' ');
end

% disp('--- Створюємо нелінійні дані ---');
function [Data_nonlin, size_new] = makeNonLinearData(Dat)
    % Ініціалізуємо масив нелінійних даних
    Data_nonlin = Dat;
    % Змінюємо її розмірність до 55 стовпчиків
```

```
    [a, b] = size(Dat.data);  % a, 10
    size_new = [a b*(b+1)/2]; % a, 55
    Data_nonlin.data = zeros(size_new);
    % Заповнюємо перші b-1 стовпчики лінійними даними незалежних змінних
    Data_nonlin.data(:,1:b-1) = Dat.data(:,1:end-1);
    % Заповнюємо решту стовпчиків нелінійними даними незалежних змінних
    k = b-1;            % 9
    for i = 1:b-1       % 9
        for j = 0:b-1-i % 9-i
            k=k+1;
            Data_nonlin.data(:,k) = Dat.data(:,i).*Dat.data(:,i+j);
        end
    end
    % Заповнюємо останній 55-ий стовпчик залежною змінною
    Data_nonlin.data(:,size_new(2)) = Dat.data(:,end);
end

% % disp('--- Стандартизація даних ---');
function Data_scaled = scaleData(Dat)
    % scaleData(Data) проводить стандартизацію даних незал. змінних
    % z = zscore(X) - стандартизує дані на основі середнього значення та
    %     стандартного відхилення вихідних даних X (z=(x-mean(X))./std(X))
    Data_scaled = Dat;
    size_dat = size(Data_scaled.data);
    Data_scaled.data(:, 1:end-1) = 0;
    Dat_mean = zeros(1, size_dat(2)); Dat_std = zeros(1, size_dat(2));
    for i = 1:size_dat(2)-1
        [Data_scaled.data(:,i),Dat_mean(i),Dat_std(i)] = ...
           zscore(Dat.data(:, i));
    end
end

% disp('--- Розбиття даних вибірки ---');
function [dataTrain, dataTest] = dataTrainAndTest(Dat)
    % dataTrainAndTest(Dat) розбиває вибірку на тренувальну та тестову
    % cvpartition(size,'HoldOut',0.3) визначає випадковий розділ набору
    %                                 даних у заданому співвідношенні
    dataTrain = Dat; dataTest  = Dat;
    % Перехресна варіація (трен: 70%, тест: 30%)
    cv = cvpartition(size(Dat.data,1),'HoldOut',0.3);
    idx = cv.test;
    % Відокремлення тренування та тестування
    dataTrain.data = Dat.data(~idx,:);
    dataTest.data  = Dat.data(idx,:);
end

% disp('--- Знаходження даних для лог. регресії ---');
function [probability, theta, cost] = funLogRegress(Dat)
    X = Dat.data(:, 1:end-1);
    y = Dat.data(:, end);

    % Налаштуємо матрицю даних належним чином: додамо одиниці для вільного члена
в р-ні регресії
    [m, n] = size(X);
    X = [ones(m, 1) X];

    % Ініціалізація параметрів оцінки
    initial_theta = zeros(n + 1, 1);

    % Обчислимо і відобразимо значення початкової функції втрат(cost) і
градієнт(grad)
    [cost, grad] = costFunction(initial_theta, X, y);
```

```
    % Встановлюємо параметри для fminunc
    options = optimset('GradObj','on','Algorithm','trust-region');

    % Запустимо fminunc, щоб отримати оптимальні коефіцієнти theta з р-ня лог
регресії
    % z = X * theta = x(1)*theta(1) + x(2)*theta(2) + ... + x(10)*theta(10)
    % Ця функція повертає theta і значення функції втрат(cost).
    [theta, cost] = ...
    fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);

    %fprintf('Кількість об`єктів у вибірці: %d\n\n', length(A.data));
    %fprintf('Функцію втрат(помилки) в theta знайдено через fminunc: %f\n\n',
cost);

    %disp('Коефіцієнти для лінійної частини логістичної регресії:');
    %fprintf('theta: \n'); fprintf(' %f \n', theta);disp(' ');

    % Обчислюємо ймовірність того чи поверне клієнт кредит
    probability = sigmoid(X * theta);

    % Обчислюємо точність на нашому наборі даних
    %prediction = predict(theta, X); % або round(probability)
end

% disp('--- Розрахування функції втрат та градієнта ---');
function [J, grad] = costFunction(theta, X, y)
    %  costFunction(theta, X, y) обчислює: J - значення функції втрат(cost)
    %  при використанні theta як параметру для логістичної регресії та
    %  grad - градієнта втрат, де X = [ones() X];
    %  theta з найменшим J є найкращою theta

    % кількість прикладів
    m = length(y);

    % градієнт
    grad = zeros(size(theta));

    % гіпотеза
    h = sigmoid(X * theta);

    J = -(1 / m) * sum( (y .* log(h)) + ((1 - y) .* log(1 - h)) );

    for i = 1 : size(theta, 1)
        grad(i) = (1 / m) * sum((h - y).* X(:,i));
    end

end

% disp('--- Розрахування сігмоїдної функції ---');
function g = sigmoid(z)
    %  Розраховуємо сигмовидну функцію
    %  g = sigmoid(z) він обчислює сигмоїду z.
    g = 1 ./ (1 + exp(-z));
end

% disp('--- Розрахування ймовірності ---');
function probability = calcProbability(theta, Dat)
    % Ініціалізація матриці даних незалежних змінних
    X = Dat.data(:, 1:end-1);

    % Налаштуємо матрицю даних належним чином:
```

```
    % додамо одиниці для вільного члена в р-ні регресії
    [m, n] = size(X);
    X = [ones(m, 1) X];

    % Обчислюємо ймовірність того чи поверне клієнт кредит
    probability = sigmoid(X * theta);
end

% disp('--- Передбачення чи повернуть кредит -> 0 або 1 ---');
function p = predict(theta, X)
    % Передбачаємо, чи буде мітка 0 чи 1, використовуючи навчені логістики
    % параметри(коефіцієнти) регресії theta
    p = round(sigmoid(X * theta));
    % p = predict(theta, X) обчислює передбачення для X за допомогою
    % порога на рівні 0,5 (тобто, якщо sigmoid(theta'*x) >= 0.5, передбачити 1)
end

% disp('--------- Друк коефіцієнтів регресії ---------');
function printThetaAndOrderedTheta(theta, countOrderTheta)
    % printThetaAndOrderedTheta(theta, countOrderTheta)
    % друкує коефіцієнтів theta та впорядковані за модулем theta,
    % де countOrderTheta - к-сть впорядкованих коеф. theta для друку

    % Друк theta
    str_koef(1:length(theta)) = "."; % 1:55
    str_koef(1:10) = ...
       ["0. константа"; "1. вік"; "2. стать"; "3. шлюб"; "4. утриманці";
        "5. дохід"; "6. досвід роботи"; "7. термін проживання";
        "8. нерухомість"; "9. місячний платіж"];
    kk = 11;
    for i = 2:10
        for j = i:10
            str_koef(kk) = str_koef(i) + " x " + str_koef (j);
            kk = kk + 1;
        end
    end
    fprintf('Коефіцієнт  №   Змінна\n\n');
    for j = 1:length(theta)
        fprintf(' ');
        if theta(j) > 0
            fprintf(' ');
        end
        fprintf('%f\t%d\t[%s]\n', theta(j), j-1, str_koef(j));
    end

    % Друк впорядкованих за модулем theta
    c = countOrderTheta;
    if countOrderTheta == 'all'
        c = length(theta);
    elseif countOrderTheta > length(theta)
        c = length(theta);
    elseif countOrderTheta < 0
        c = 0;
    end

    abs_theta = abs(theta); abs_theta0 = abs_theta;
    fprintf('\nКоефіцієнти(theta) впорядковані за модулем(перші %d з %d шт):\n',
...
            c, length(theta));
    fprintf('\nКоефіцієнт  Змінна\n\n');
    for i = 1:c
        mx = max(abs_theta0);
```

```
        ind1 = find(abs_theta == mx);
        fprintf(' ');
        if(theta(ind1) > 0)
            fprintf(' ');
        end
        fprintf('%f  [%s]\n', theta(ind1), str_koef(ind1));
        abs_theta0(ind1) = 0;
    end
end

% disp('--------- Розрахування accuracy ---------');
function accuracy = calcAccuracy(Dat, probability)
    % Розрахування accuracy (частки правильних відповідей )
    % 2 способи обчислення:
    % accuracy_1 = (TP+TN)/(TP+TN+FP+FN)(через матрицю помилок)
    % accuracy_2 = sum(OrigValues == PredValues) / numel(PredValues)
                % Функція numel(A) повертає кількість елементів у масиві A.
    OrigValues = Dat.data(:, end);
    PredValues = round(probability);
    accuracy = sum(OrigValues == PredValues) / numel(PredValues);
end

% disp('--------- Розрахування precision та recall---------');
function [precision, recall] = calcPrecisionAndRecall(Dat, probability)
    % precision (точність)
    % recall (повнота)

    % Розрахування confusion matrix (матриці помилок)
    %            y = 1                 y = 0
    % y' = 1    True Positive  (TP)    False Positive (FP)
    % y' = 0    False Negative (FN)    True Negative (TN)

    TP = 0; FN = 0; FP = 0; TN = 0;
    for i = 1:length(Dat.data)
        if Dat.data(i, size(Dat.data, 2)) == 1
            if probability(i) >= 0.5
                TP = TP + 1;
            else
                FN = FN + 1;
            end
        else
            if probability(i) >= 0.5
                FP = FP + 1;
            else
                TN = TN + 1;
            end
        end
    end
    %disp('confusion matrix (матриця помилок):');disp(' ');
    %fprintf('TP: %d\t FP: %d\nFN: %d\t TN: %d\n',TP, FP, FN, TN);
    %fprintf('Їх сума(має = %d): %d\n\n', length(A.data), TP+FP+FN+TN);

    precision = TP / (TP + FP);
    recall = TP / (TP + FN);
end

% disp('------------- Побудова графіку ROC-кривої -------------');
function plotROC(X, Y, AUC)
    disp('Побудова графіку ROC-кривої...');
    % Будуємо ROC-криву разом із прямою y = x.
    figure;
    plot(X, Y);   hold on;
```

```
    xx = [0 1]; yy = [0 1];
    plot(xx, yy, 'r-.', 'linewidth', 1);
    xlabel('FPR'); % False Positive Rate
    ylabel('TPR'); % True Positive Rate
    title(['ROC крива (AUC = ' num2str(AUC) ' )']);
    legend('ROC крива', 'y = x');
end
```