Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра математичного та комп'ютерного моделювання

Дипломна робота

бакалавра

на тему: **«Розв'язання задачі кредитування за допомогою нейромережевих технологій»**

«Solving the problem of lending with the help of neural network technologies»

Виконав: студент денної форми навчання
спеціальності 113 Прикладна математика
Крутоголов Данило Геннадійович

Керівник: канд. фіз.-мат. наук, доц. Вербіцький В. В.
Рецензент: канд. фіз.-мат. наук, доц. Таірова М. С.

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від «_____» _____ 2021 р.
Завідувач кафедри

_____ _____

Захищено на засіданні ЕК № _____
Протокол № ___ від «___» ___ 2021 р.
Оцінка _____ / _____ / _____
Голова ЕК

_____ _____

Одеса — 2021 р.

ЗМІСТ

ВСТУП

Банківська система України в сучасних умовах розвивається динамічно та поступово інтегрується в світовий банківський простір. Цей процес постійно потребує ефективного механізму управління банківською діяльністю та дієвої системи регулювання і контролю банківських операцій. В умовах загострення конкуренції між банками та стрімкого розвитку інформаційних банківських технологій, управління банківською діяльністю, потребує кваліфікованого менеджерського персоналу. Тому є необхідність у постійному вивченні та дослідженні нових способів та підходів до здійснення банківських операцій.

Сьогодні, обговорення нейронних мереж відбуваються всюди. Перспектива їх використання здається досить яскравою, в світлі рішення нетрадиційних проблем і є ключем до цілої технології. На даний час більшість розробок нейронних мереж принципово працюють, але можуть існувати процесорні обмеження. [10] Дослідження спрямовані на програмні і апаратні реалізації нейромереж. Компанії працюють над створенням трьох типів нейрочипів: цифрових, аналогових і оптичних, які обіцяють бути хвилею близького майбутнього. [8]

Об'єктом дослідження роботи є задача видачі кредиту.

Предметом дослідження є нейромережеві технології.

Мета даної роботи - побудова ефективної нейромережі для розв'язання задачі кредитування.

РОЗДІЛ 1

ПОСТАНОВА ЗАДАЧІ

Вирішити питання «яким клієнтам можна видавати кредит, а яким ні?» використовуючи нейронну мережу при малій кількості навчальних даних, а також покращити модель за допомогою знаходження та видалення даних, які не впливають на кінцевий результат. Навчальним набором даних виступає база даних, яка містить інформацію про клієнтів, зокрема: ”Сума кредиту”, ”Термін кредиту”, ”Мета кредитування”, ”Вік”, ”Стать”, ”Освіта”, ”Приватна властність”, ”Квартира”, ”Площа квартири” та інші. Необхідно побудувати модель, за допомогою якої буде проаналізовано базу даних клієнтів використовуючи метод ”Machine Learning” на основі нейросимулятора аналітичного пакету ”Deductor” та порівняти її результати з нейромережею розробленою використовуючи метод ”Deep Learning” та програмну мову Python з бібліотекою Tensorflow. [11] [7] Вирішити проблему перенавчання моделі.

РОЗДІЛ 2

ОСНОВНІ ПОНЯТТЯ ТА ВИЗНАЧЕННЯ НЕЙРОННИХ МЕРЕЖ

Нехай X — простір об'єктів; Y - безліч допустимих відповідей; $y^*: X \to Y$ — цільова залежність, відома тільки на об'єктах навчальної вибірки $X^l = (x_i, y_i)_{i=1}^l, y_i = y^*(x_i)$. Потрібно побудувати алгоритм $a: X \to Y$, аппроксимуючий цільову залежність y^* на всьому безлічі X. Будемо припускати, що об'єкти описуються n числовими ознаками $f_j: X \to R, j = 1, ..., n$. Вектор $(f_1(x), ..., f_n(x)) \in R_n$ називається ознаковим описом об'єкта x. [2]

Нейрон

Структурною одиницею нейронної мережі є нейрон. На вхід нейрона подається n-мірний вектор простору ознак опису $x = (x_1, ..., x_n)$. Кожному входу відповідає ваговий коефіцієнт — $\omega_1, ...\omega_n$. Виходом нейрона є значення функції активації від зваженої суми вхідних значень. Також на вхід подається вільний член, який має постійне значення, та також має вагу.

$$\alpha(x) = \varphi\left(\sum_{j=1}^{n} \omega_j x^i\right) = \varphi(\langle\omega, x\rangle)$$

Даний вид нейрона є найпростішим лінійним класифікатором і може самостійно навчатися. Якщо нейрон знаходиться в прихованому або вихідному шарі нейронної мережі, то на вхід йому подаються вихідні значення інших нейронів. Поточний стан нейрона визначається, як зважена сума його входів: $s = \sum_{i=1}^{n} x_i \cdot \omega_i$. [4]

Нейронна мережа

З'єднувати нейрони в мережу можна багатьма способами. Розглянемо найбільш поширений і досліджений спосіб композиції нейронів - багатошаровий персептрон. Нейрони розташовуються шарами, при цьому кожен нейрон

з наступного шару пов'язаний з усіма нейронами попереднього. Виділяють три типи шару - вхідний шар неройнов, прихований шар і вихідний шар. Найчастіше використовується тришаровий персептрон, тобто з одним прихованим шаром. По теоремі Колмогорова мережу такий структури з нелінійної функцією активації здатна апроксимувати будь-яку функцію. Вихідні значення алгоритму знімаються з виходів останнього шару нейронної мережі. Кількість нейронів у вихідному шарі відповідає розмірності вектора відповідей. Кількість нейронів у вхідному шарі відповідає розмірності простору ознак опису об'єкта. Кількість нейронів у прихованих шарах вибирається експертом чи за допомогою евристик щодо оптимізації структури мережі.

Навчання штучних нейронних мереж

Правило навчання або алгоритм навчання - це метод або математична модель, яка підвищує продуктивність штучної нейронної мережі і, як правило, це правило застосовується багаторазово по цілій мережі. Це робиться шляхом поновлення ваг і рівнів упередженості мережі. Правило навчання може прийняти існуючі умови (ваги і зміщення) мережі і порівнює очікуваний результат і фактичний результат мережі, щоб дати нові та вдосконалені значення для ваг і зсуву. В залежності від складності конкретної моделі, яка моделюється, правило навчання мережі може бути настільки ж просто, як XOR входів або середньоквадратичної помилки, або це може бути результатом кількох диференціальних рівнянь. Правило навчання є одним з факторів, який визначає, як швидко або наскільки точні можуть бути розроблені штучні мережі. [3]

Парадигми навчання

На сьогодні відомо три парадигми навчання нейронних мереж, в основу яких покладено особливості машинного навчання:

Навчання з вчителем (supervised learning)

Передбачає, що для кожного вхідного вектора x_i існує вектор вихідних значень d_i. Разом ці два вектора називають навчальною парою (x_i, d_i), а множину навчальних пар — навчальною вибіркою. Процес навчання зводиться до почергового подавання на вхід нейронної мережі навчальних пар, вираховування похибки між дійсним і бажаним значенням нейронної $\delta = y - d$ та корегування параметрів мережі в бік зменшення цієї похибки. [3]

Навчання без вчителя (unsupervised learning)

Один зі способів машинного навчання, при вирішенні яких випробовувана система спонтанно навчається виконувати поставлене завдання, без втручання з боку експериментатора. З точки зору кібернетики, є одним з видів кібернетичного експерименту. Як правило, це підходить тільки для задач, в яких відомий опис множини об'єктів (навчальна вибірка), і необхідно виявити внутрішні взаємозв'язки, залежності, закономірності, що існують між об'єктами. [3]

Навчання з підкріпленням

Проміжний варіант двох попередніх парадигм. Замість «вчителя» в схему навчання вводиться блок «критика», який відслідковує реакцію середовища на вхідний сигнал і опираючись на неї визначає евристичну похибку, яку покладено в процес навчання мережі. Вказані парадигми базуються на відповідних правилах навчання, які визначають основні особливості їх застосування. [3]

Правила навчання

Теорія навчання розглядає три фундаментальні властивості, пов'язані з навчанням за прикладами: ємність, складність зразків і обчислювальна складність. Під ємністю розуміється, скільки зразків може запам'ятати ме-

режа, і які функції і межі ухвалення рішень можуть бути на ній сформовані. Складність зразків визначає число навчальних прикладів, необхідних для досягнення здатності мережі до узагальнення. Дуже мале число прикладів може викликати «перенавчання» мережі, коли вона добре функціонує на прикладах навчальної вибірки, але погано — на тестових прикладах, які підлягають тому ж статистичному розподілу. Відомі 4 основних типи правил навчання: корекція за помилкою, навчання Больцмана, Дельта правило і навчання методом змагання. Ці правила є базовими і основними для навчання ШНМ, всі решта правила які існують є модифікаціями цих правил. [5]

Правило корекції за помилкою

При навчанні з вчителем для кожного вхідного прикладу заданий бажаний вихід d. Реальний вихід мережі у може не збігатися з бажаним. Являє собою такий метод навчання, при якому вага зв'язку не змінюється до тих пір, поки поточна реакція перцептрона залишається правильною. При появі неправильної реакції вага змінюється на одиницю, а знак + чи − визначається протилежним від знаку помилки. Навчання має місце тільки у разі, коли перцептрон помиляється. Відомі різні модифікації цього алгоритму навчання: метод корекції помилок без квантування, метод корекції помилок з квантуванням, метод корекції помилок з випадковим знаком підкріплення, метод корекції помилок з випадковими збуреннями. [5]

Правило навчання Больцмана

Правило навчання Больцмана є стохастичним правилом навчання, яке виходить з інформаційних теоретичних і термодинамічних принципів. Метою навчання Больцмана є таке налаштування вагових коефіцієнтів, при якому стани видимих нейронів задовольняють бажаний розподіл вірогідності. Навчання Больцмана може розглядатися як спеціальний випадок корекції за помилкою, в якому під помилкою розуміється розбіжність кореляцій полягань в двох режимах. [5]

Основні ознаки навчання:

- Використовують каскад багатьох шарів вузлів нелінійної обробки для виділяння ознак та перетворення. Кожен наступний шар використовує вихід із попереднього шару як вхід. Алгоритми можуть бути з керованим або спонтанним навчанням, а застосування включають розпізнавання образів (спонтанне) та класифікацію (керовану).
- Ґрунтуються на навчанні (спонтанному) декількох шарів ознак або представлень даних. Ознаки вищих рівнів виводяться з ознак нижчих рівнів для формування ієрархічного представлення.
- Є частиною ширшої області машинного навчання з навчання представлень даних.
- Навчаються кільком рівням представлень, що відповідають різним рівням абстракції; ці рівні формують ієрархію понять.

Дельта правило

Власне дельта-правилом називають математичну, трохи загальнішу форму запису правил Хебба. Нехай вектор $X = x_1,x_2,.....,x_r,....,x_m$ — вектор вхідних сигналів, а вектор $D = d_1,d_2,...,d_k,...,d_n$ $D = d_1,d_2,...,d_r,...,d_m$ — вектор сигналів, які повинні бути отримані від перцептрона під впливом вхідного вектора. Тут — кількість нейронів, що входять до перцептрону. Вхідні сигнали, поступивши на входи перцептрону, були зважені і підсумовані, в результаті чого отримано вектор $Y = y_1,y_2,...,y_r,...,y_m$ вихідних значень перцептрона. Тоді можна визначити вектор помилки $E = e_1,e_2,...,e_r,...,e_m$, розмірність якого збігається розмірністю вектором вихідних сигналів. Компоненти вектора помилок визначаються як різниця між очікуваним і реальним значенням вихідного сигналу нейрону перцептрона:

$$E = D - Y$$

За таких позначеннях формулу для коригування j-ї ваги i-го нейрона можна записати так:

$$w_j(t+1) = w_j(t) + e_i x_j$$

Номер сигналу j змінюється в межах від одиниці до розмірності вхідного вектора m. Номер нейрону i змінюється в межах від одиниці до кількості

нейронів n. Величина t — номер поточної ітерації навчання. Таким чином, вага вхідного сигналу нейрона змінюється у бік зменшення помилки пропорційно величині сумарної помилки нейрона. Часто вводять коефіцієнт пропорційності η , на який множиться величина помилки. Цей коефіцієнт називають швидкістю навчання. Таким чином, підсумкова формула для коректування ваг зв'язків перцептрону має наступний вигляд:

$$w_j(t+1) = w_j(t) + e_i \eta x_j w_j(t+1) = w_j(t) + e_i \eta x_j \text{ [9]}$$

Навчання методом змагання

На відміну від навчання Хебба, в якому безліч вихідних нейронів можуть збуджуватися одночасно, при навчанні змагання вихідні нейрони змагаються між собою за активізацію. Це явище відоме як правило «переможець бере все». Подібне навчання має місце в біологічних нейронних мережах. Навчання за допомогою змагання дозволяє кластеризувати вхідні дані: подібні приклади групуються мережею відповідно до кореляцій і подаються одним елементом. При навчанні модифікуються тільки ваги нейрона, що «переміг». Ефект цього правила досягається за рахунок такої зміни збереженого в мережі зразка (вектора вагів зв'язків нейрона, що «переміг»), при якому він стає трохи ближчим до вхідного прикладу

Алгоритм навчання

Взагалі є щонайменш 3 алгоритма навчання нейронних мереж, але ми розглянемо тільки один

Алгоритм зворотнього поширення

Зворотне поширення — це метод обчислення градієнту функції втрат (видає витрати, пов'язані з заданим станом) по відношенню до ваг в ШНМ. Основи неперервного зворотного поширення було виведено в контексті теорії керування Келлі 1960 року та Брайсоном 1961 року з використанням принципів динамічного програмування. 1962 року Дрейфус опублікував простіше виведення, засноване лише на ланцюговому правилі. Брайсон та

Хо описали його як метод багатоетапної оптимізації динамічних систем 1969 року. 1970 року Ліннаінмаа остаточно опублікував загальний метод автоматичного диференціювання (АД) дискретних зв'язних мереж вкладених диференційовних функцій. Він відповідає сучасному баченню зворотного поширення, яке є ефективним навіть коли мережі є розрідженими. 1973 року Дрейфус застосував зворотне поширення для пристосування параметрів контролерів пропорційно градієнтам похибок. 1974 року Вербос зазначив можливість застосування цього принципу до ШНМ, і 1982 року він застосував метод АД Ліннаінмаа до нейронних мереж способом, який широко застосовується сьогодні. 1986 року Румельхарт, Хінтон та Вільямс зазначили, що цей метод може породжувати корисні внутрішні представлення вхідних даних в прихованих шарах нейронних мереж. 1993 року Ван став першим переможцем міжнародного змагання з розпізнавання образів за допомогою зворотного поширення. Уточнення ваг зворотного поширення можливо здійснювати за допомогою стохастичного градієнтного спуску із застосуванням наступного рівняння:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}} + \xi(t)$$

де η є темпом навчання, C є функцією витрат (втрат), а $\xi(t)$ — стохастичним членом. Вибір функції витрат залежить від таких чинників як тип навчання (кероване, спонтанне, з підкріпленням тощо) та функції збудження. Наприклад, при здійсненні керованого навчання на задачі багатокласової класифікації поширеними варіантами вибору функції збудження та функції витрат є нормована експоненційна функція та функція перехресної ентропії відповідно. Нормалізовану експоненційну функцію визначають як $p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}$, p_j представляє ймовірність класу (вихід вузла j, а x_j та x_k представляють загальний вхідний сигнал вузлів j та k одного й того ж рівня відповідно. Перехресну ентропію визначають як $C = -\sum_j d_j \log(p_j)$, де d_j представляє цільову ймовірність для вузла виходу j, а p_j є виходом ймовірності для j після застосування функції збудження.

Це можливо використовувати для виведення обмежувальних коробок об'єкта у вигляді двійкової маски. Їх також використовують для багатомасштабної регресії для підвищення точності визначення положення. Регресія

на основі ГНМ може навчатися ознак, що схоплюють геометричну інформацію, на додачу до того, що вони слугують добрим класифікатором. Вони усувають вимогу явного моделювання частин та їхніх взаємозв'язків. Це допомагає розширити розмаїття об'єктів, яких можна навчитися. Модель складається з декількох шарів, кожен з яких має випрямляч як функцію збудження для нелінійного перетворення. Деякі шари є згортковими, тоді як деякі є повнозв'язними. Кожен згортковий шар має додаткове максимізаційне агрегування. Мережу тренують для зведення до мінімуму похибки L^2 для передбачування маски, що пробігає весь тренувальний набір, що містить обмежувальні коробки, представлені як маски. До альтернатив зворотному поширенню належать машини екстремального навчання, «безпоширні» (англ. «No-prop») мережі, тренування без пошуку з вертанням, «безвагові» (англ. weightless) мережі та не-конективістські нейронні мережі. [10]

РОЗДІЛ 3

РОЗВ'ЯЗАННЯ ЗАДАЧІ КРЕДИТУВАННЯ ЗА ДОПОМОГУЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

Вирішити питання «яким клієнтам можна видавати кредит, а яким ні?» використовуючи нейронну мережу при малій кількості навчальних даних, а також покращити модель за допомогою знаходження та видалення даних, які не впливають на кінцевий результат. Навчальним набором даних виступає база даних, яка містить інформацію про клієнтів, зокрема: "Сума кредиту", "Термін кредиту", "Мета кредитування", "Вік", "Стать", "Освіта", "Приватна властність", "Квартира", "Площа квартири". Необхідно побудувати модель, за допомогою якої буде проаналізовано базу даних клієнтів використовуючи метод "Machine Learning" на основі нейросимулятора аналітичного пакету "Deductor" та порівняти її результати з нейромережею розробленою використовуючи метод "Deep Learning" та програмну мову Python з бібліотекою Tensorflow. Вирішити проблему перенавчання моделі.

Розв'язання задачі за допомогою нейросимулятора

Будемо рухатись по такій послідовності дій:

1) Створення нейросимулятора.
2) Навчання мережі.
3) Отримання результату.
4) Розробка нейромережі використовуючи Tensorflow.
5) Навчання мережі.
6) Вирішення проблеми перенавчання.
7) Оптимізація датасету та перенавчання.
8) Порівняння результатів.

Розглянемо рішення задачі "Можливість видачі кредиту" з використанням нейросимулятора в аналітичному пакеті Deductor (BaseGroup). [6] Deductor — аналітичний пакет, який надає самонавчальні алгоритми і машинне навча-

ння: дерева рішень, нейронні мережі, карти, що самоорганізуються, асоціативні правила. Аналіз тимчасових рядів: виявлення сезонності, тренду і випадкової складової. Безліч способів оцінки якості моделей з можливістю вибору кращої. Спеціалізовану візуалізацію, що полегшує інтерпретацію і підвищує довіру до результатів. Завдання о кредитуванні відноситься до групи завдань класифікації, тобто навчання з учителем.

Вхідні данні статистично виглядають так (см Рис. 3.1):

	Метка столбца	Статистика: Кол-во значений = 149								
		Гистогра...	Мини...	Макс...	Сред...	Стан...	Сумма	Сумм...	Кол-в...	Кол-в...
1	9.0 Сумма кредита		2000	69500	3805,36913	5428,64038	3547000	1,19668E11		0
2	9.0 Стоимость кред...		400	13900	761,073826	085,728076	709400	786720000		0
3	9.0 Срок кредита		6	48	4,97986577	,269588316	2232	46152		0
4	Дата кредитова...		01.01.2003	12.01.2003	003 5:47:55	ин 07:49:33				0
5	ab Цель кредитова...		4	33	19,53	9,586	2910	70434	6	0
6	9.0 Количество		1	1	1	0	149	149		0
7	9.0 Возраст		19	70	4,96644295	2,97316154	5210	207084		0
8	ab Пол		3	3	3	0	447	1341	2	0
9	ab Образование		6	11	9,154	2,336	1364	13294	3	0
10	ab Частная собств...		2	3	2,289	0,455	341	811	2	0
11	ab Квартира		2	3	2,409	0,493	359	901	2	0
12	9.0 Площадь кварт...		12	70	2,63087248	1,89103057	4862	179578		0
13	ab Способ приобре...		6	25	8,168	5,667	1217	14693	3	0
14	ab Расположение		5	7	5,55	0,896	827	4709	2	0
15	ab Машина		9	14	12,342	1,758	1839	23155	3	0
16	9.0 Срок эксплуата...		1	14	,651006711	,979493369	693	4537		0

Рис. 3.1. Вигляд статистичних даних

Спершу розіб'ємо вхідну множину даних на "обучающее"і "тестовое". Спосіб розбиття вхідної множини даних за замовчуванням заданий "Случайно"(см Рис. 3.2). Це означає, що 95% даних будуть використані для навчання нейронної мережі, а інші 5% для тесту.

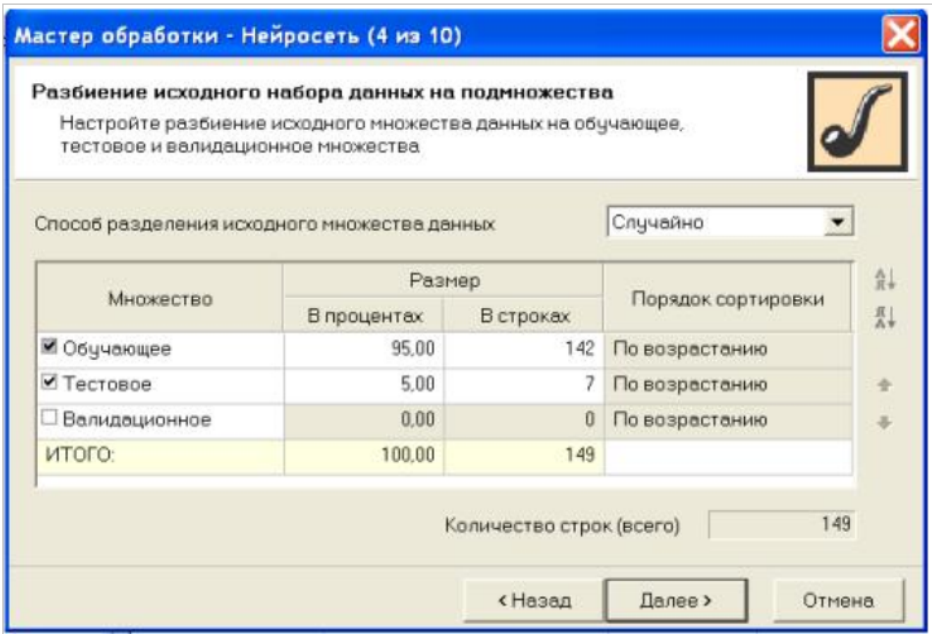


Рис. 3.2. Розбиття вхідного набору даних на підмножини

Далі необхідно вибрати алгоритм і параметри навчання нейронної мережі (см Рис. 3.3).

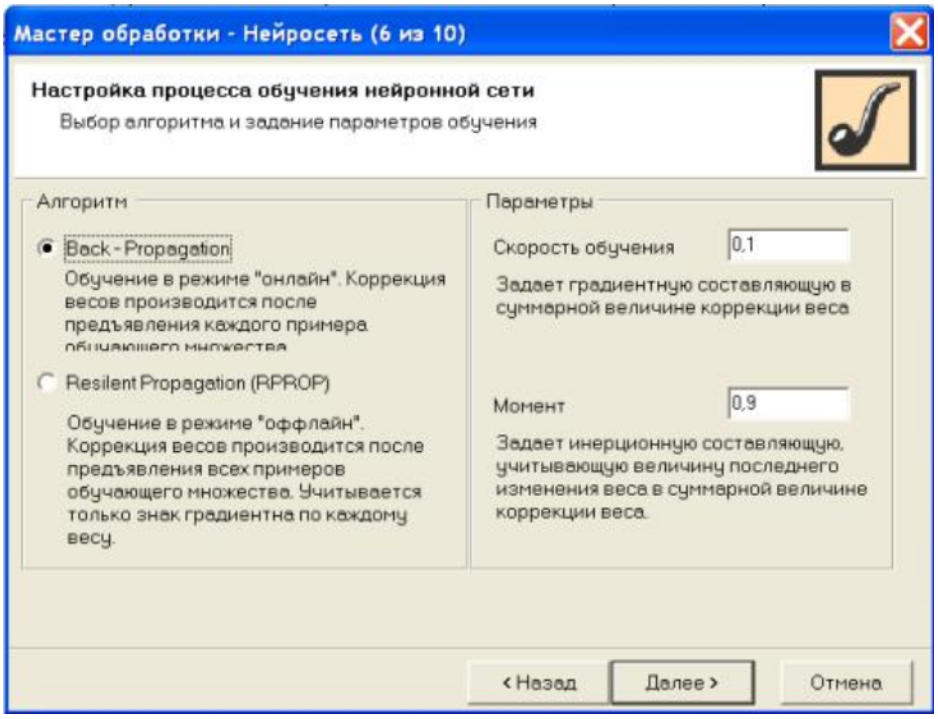


Рис. 3.3. Налаштування процесу навчання даних нейронної мережі

На наступному кроці налаштовуємо умови зупинки навчання (см Рис. 3.4). Будемо вважати приклад розпізнаним, якщо помилка менше 0,005, і вкажемо умови зупинки навчання при досягненні епохи 10000. На наступному кроці запускаємо процес навчання і спостерігаємо за зміною

величини помилки і відсотку розпізнаних прикладів в навчальній та тестовій множині. У нашому випадку ми бачимо, що на епосі № 4536 в навчальній множині розпізнано 83,10% прикладів, а на тестовому - 85,71% прикладів. Після закінчення процесу навчання для інтерпретації отриманих результатів ми маємо можливість вибрати візуалізатори зі списку запропонованих.

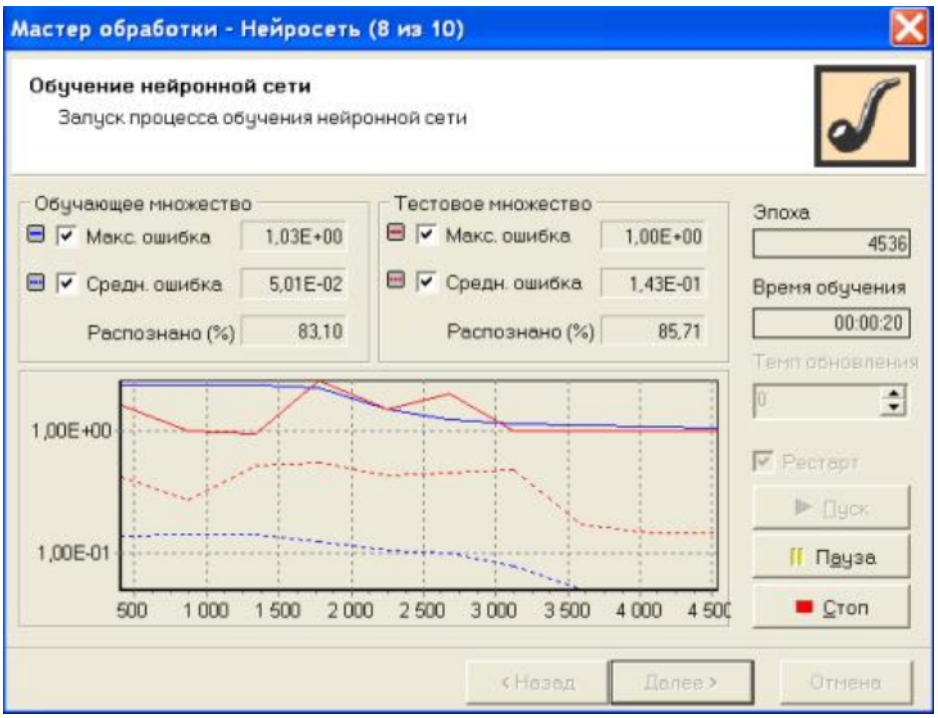


Рис. 3.4. Налаштування умови зупинки навчання

Нижче продемонстрована таблиця сполученості. На її діагоналі розташовані приклади, які були правильно розпізнані, тобто 55 клієнтів, яким можна видавати кредит, і 89 клієнтів, яким видавати кредит не варто. В інших осередках розташовані ті клієнти, які були віднесені до іншого класу (1 і 4). Можна вважати, що правильно класифіковані практично всі приклади – 96,64%.

Таблиця сполученості має вигляд(см Рис. 3.5):

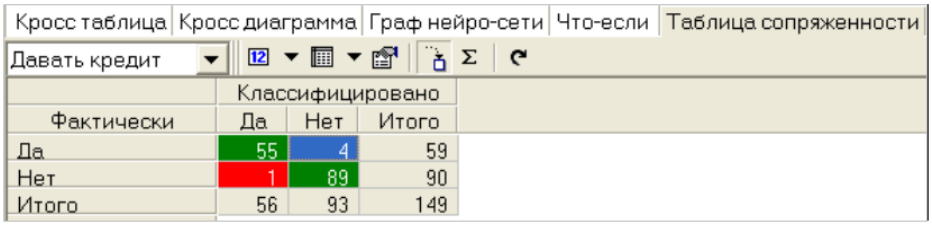


Рис. 3.5. Результати навчання

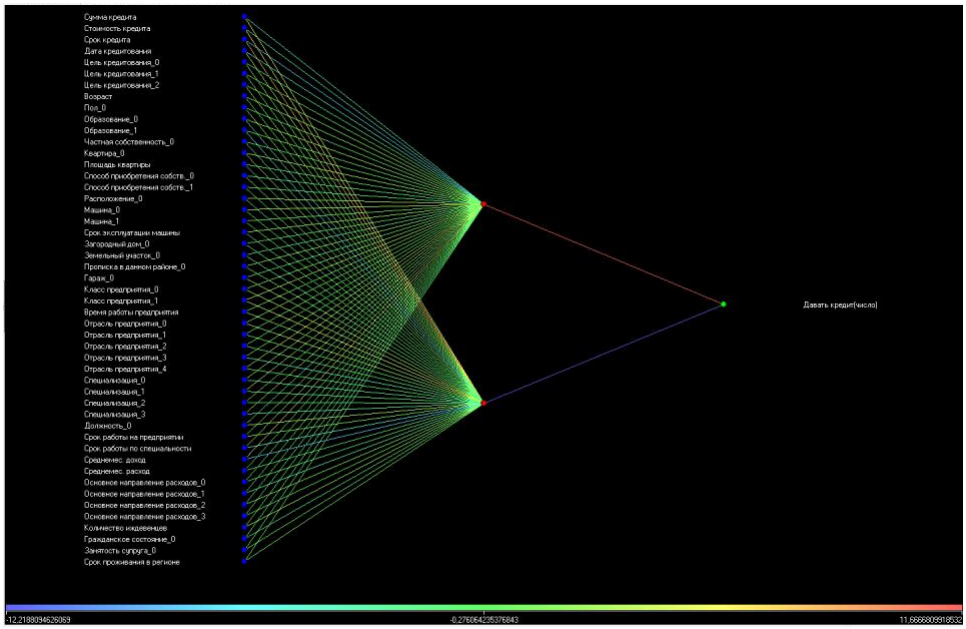
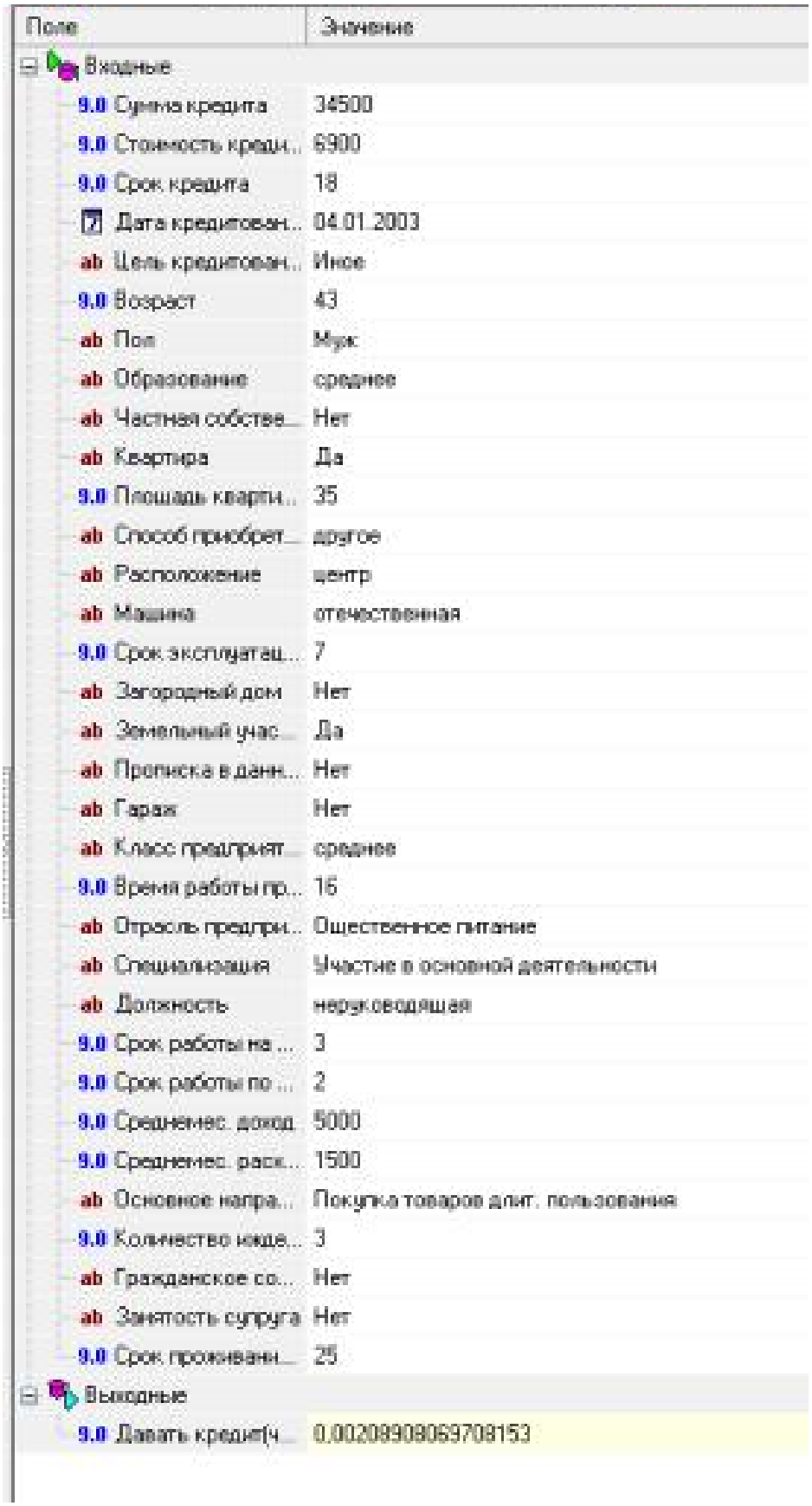


Рис. 3.6. Граф нейромережі з кореляцією

Проаналізувавши графіки, можна побачити, що найчутливішими та найважливішими стали поля ”Срок кредита”, ”Образование”, ”Срок работы по специальности” та ”Среднемесячный доход” (см Рис. 3.6). Це означає, що змінюючи ці поля, вихідні данні будуть змінюватись найбільше. Наприклад, візьмемо та протестуємо нейронну мережу на нових вхідних данних. Запишемо в поля ”Срок кредита” 18, ”Образование” среднее, ”Срок работы по специальности” - 2, ”Среднемесячный доход” 5000. Отримаємо вихідні данні - 0.002, тобто - ні. Тепер змінемо ”Среднемесячный доход” до 10000. Отримали 0.999, тобто так, при цьому, решта грошей людини враховуючи поле ”Среднемесячный расход” змінилась з 3500 до 8500. Здається, що нейронна мережа дійсно видає хороший результат, але спробуємо підвищити ”Среднемесячный расход” до 10000. При решті грошей 0, отримали результат 0.998, гірший за той, коли решта грошей була максимальною, але й набагато кращій за той, коли решта була 3500.



Поле	Значение
Входные	
9.0 Сумма кредита	34500
9.0 Стоимость креди...	6900
9.0 Срок кредита	18
Дата кредитован...	04.01.2003
ab Цель кредитован...	Иное
9.0 Возраст	43
ab Пол	Муж
ab Образование	среднее
ab Частная собстве...	Нет
ab Квартира	Да
9.0 Площадь кварти...	35
ab Способ приобрет...	другое
ab Расположение	центр
ab Машина	отечественная
9.0 Срок эксплуатац...	7
ab Загородный дом	Нет
ab Земельный учас...	Да
ab Прописка в данн...	Нет
ab Гараж	Нет
ab Класс предприят...	среднее
9.0 Время работы пр...	16
ab Отрасль предпри...	Общественное питание
ab Специализация	Участие в основной деятельности
ab Должность	неруководящая
9.0 Срок работы на ...	3
9.0 Срок работы по ...	2
9.0 Среднемес. доход	5000
9.0 Среднемес. расх...	1500
ab Основное направ...	Покупка товаров длит. пользования
9.0 Количество кредс...	3
ab Гражданское со...	Нет
ab Занятость супруга	Нет
9.0 Срок проживани...	25
Выходные	
9.0 Давать кредит(ч...	0.002089080697091 53

Рис. 3.7. Вхідні данні

Рис. 3.8. Збільшення значення поля середньомісячного доходу

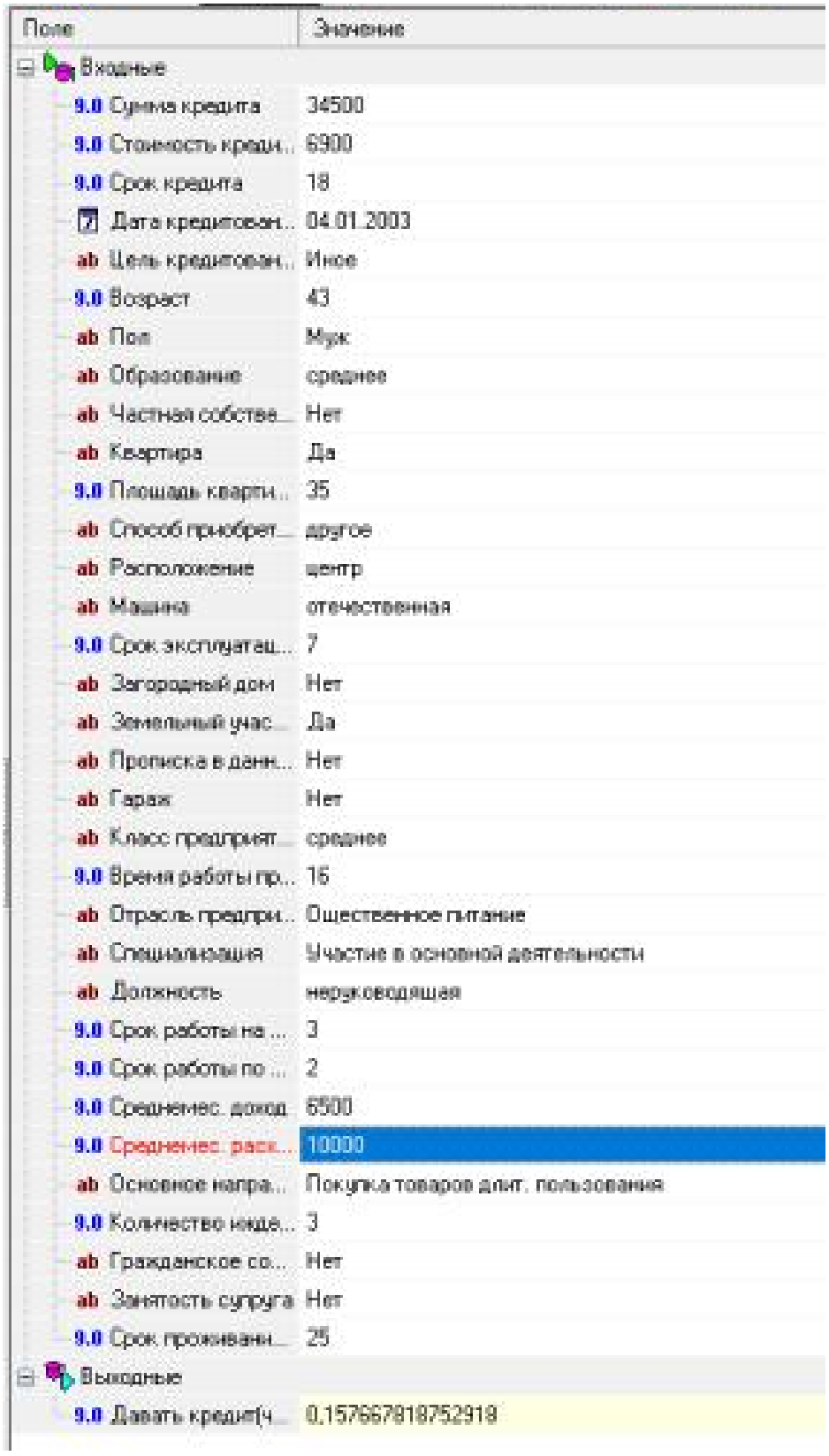


Рис. 3.9. Збільшення значення поля середньомісячних витрат так, щоб вони перевищували середньомісячний дохід

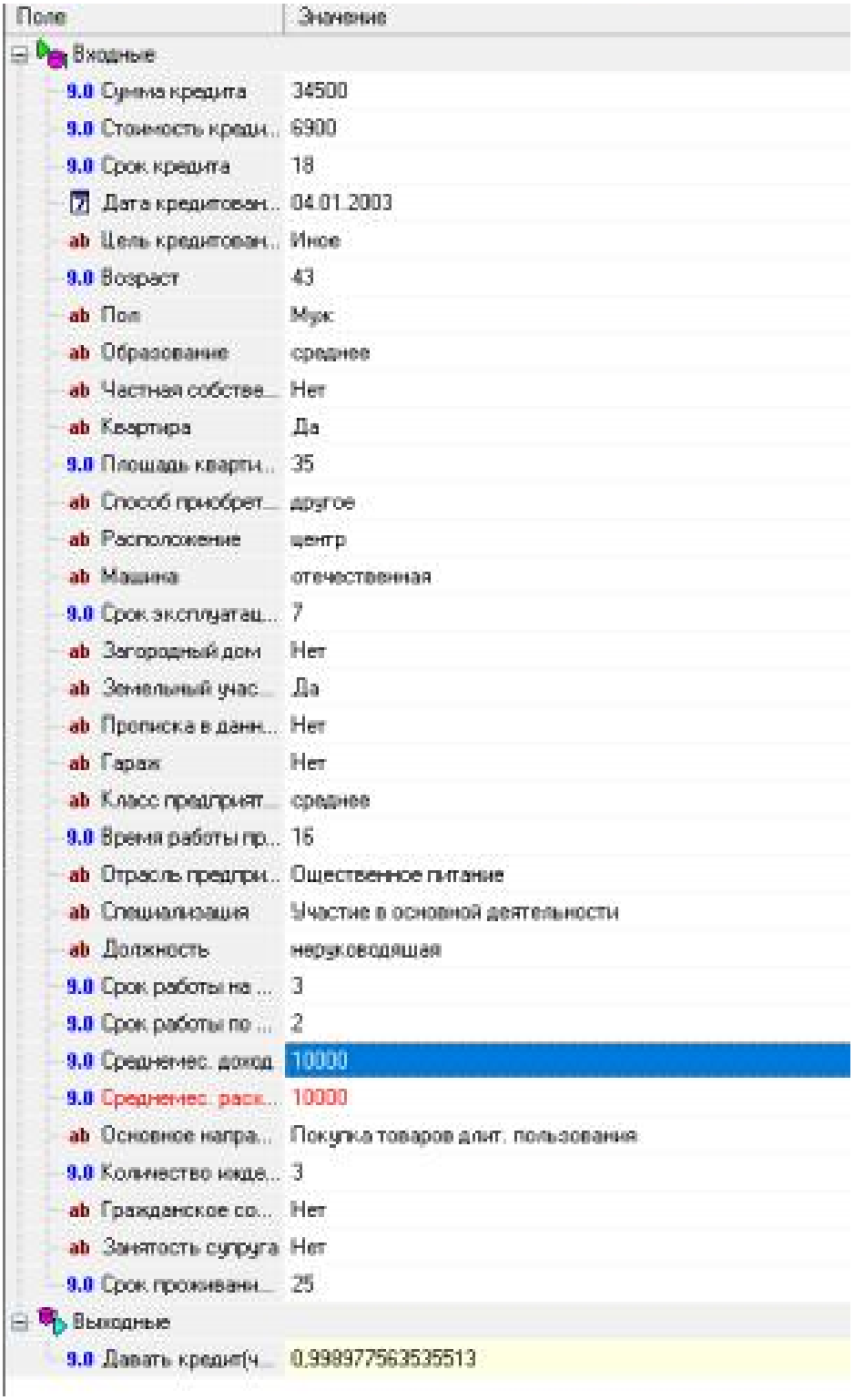


Рис. 3.10. Змінення даних середньомісячного доходу так, щоб різниця між ним і середньомісячними витратами давала 0

Розв'язання задачі за допомогою нейромережі

Розв'яжемо цю задачу за допомогою нейромережі. Будемо використовувати програмну мову "Python метод deep learning та бібліотеку "Tensorflow"для розробки моделі на тих же даних.

Поперше класифікуємо дані за чотирма категоріями:

1) Кредит – Сума кредиту, Вартість кредиту, Термін кредиту, Дата кредитування, Мета кредитування.
2) Персональні дані – Вік, Термін проживання в регіоні, Стать, Освіта, Розташування, Спеціалізація, Кількість утриманців, Прописка в даному районі, Посада, Цивільний стан, Зайнятість чоловіка або жінки.
3) Майно – Площа квартири, Термін експлуатації машини, Спосіб придбання власності, Машина, Основний напрямок витрат, Приватна власність, Квартира, Заміський будинок, Земельна ділянка, Гараж.
4) Робота – Час роботи підприємства, Термін роботи на підприємстві, Термін роботи за фахом, Середньомісячний дохід, Середньомісячні витрати, Клас підприємства, Галузь підприємства.

Це потрібно для порівняння з машинною класифікації моделі, яку модель генерує сама, а також для подальшої зміни датасету.

Далі розіб'ємо вхідні дані на ті, які потрібно класифікувати, нормалізувати та видалити(см Рис. 3.11). За це в програмі відповідають змінні to_classify, to_zero_or_one, to_drop. To_classify буде розбивати дані типу "Місце роботи" на множину, щоб модель могла видати булеве значення (у виді цифр 0 та 1) до кожної людини. На прикладі "Місце роботи" отримаємо колонки виду "Місце роботи Фірма "гудтех"', "Місце роботи Національний університет", тощо. Для різних людей значення будуть 0, якщо вони там не працюють, та 1, відповідно, якщо працюють. Змінна to_zero_or_one форматує дані типу "Так", "Ні" на "0" та "1". To_drop відповідає за дані, які будуть відкинуті. Усі інші дані будуть нормалізовані, тобто розбиті таким чином, що на виході маємо 0 чи 1. Це досягається знаходженням мінімального значення в даному стовпцю, максимального та середнього. Усі значення вище середнього автоматично стають 1, а інші, відповідно, 0.

```
import data_improvements as di
import pandas as pd

data = pd.read_csv(
    "~/Desktop/diploma/CorruptionDetector/data/credit_data.csv", sep=";")
ind = 141

to_classify = ["Пол",
               "Машина", "Класс предприятия",
               "Количество иждевенцев"]

to_zero_or_one = ["Частная собственность", "Квартира", "Загородный дом",
                  "Прописка в данном районе", "Гараж", "Должность",
                  "Гражданское состояние", "Занятость супруга"]


to_drop = ["Давать кредит", "Сумма кредита", "Срок кредита", "Дата кредитования",
           "Цель кредитования", "Образование", "Площадь квартиры", "Способ приобретения собств
           "Расположение", "Срок эксплуатации машины", "Земельный участок", "Отрасль предприят
           "Специализация", "Среднемес. доход", "Основное направление расходов"]
```

Рис. 3.11. Розбиття вхідного набору даних

Проходимо по всім даним та зберігаємо результат в форматі .csv з відношенням 95% до 5%, щоб тестувати модель на 5% даних (см Рис. 3.12).

```
for i in to_drop:
    data = data.drop(i, axis=1)

for i in to_classify:
    print(1, i)
    data = di.classify(data, i)

for i in di.nan_columns(data):
    print(2, i)
    data = di.denanization(data, i)

for i in to_zero_or_one:
    print(3, i)
    data = di.one_or_null(data, i, data[i].iloc[0]).drop(i, axis=1)

for i in data.columns:
    print(4, i)
    data = di.normalize(data, i)

data = di.compare(data)

train_data_converted = data[:ind]
test_data_converted = data[ind:]

print(train_data_converted)
print(test_data_converted)

train_data_converted.to_csv("train_deleted_high.csv")
test_data_converted.to_csv("test_deleted_high.csv")
```

Рис. 3.12. Модифікація вхідного набору даних у подання, прийнятне для навчання моделі

Будуємо модель з 5 слоями. Перший слой - вхідний із 128 нейронами, три слої приховані, останній - вихідний. Шар Dense реалізує операцію:

output = activation (dot(input, kernel) + bias) де activation функція активації, передана в якості activation аргументу, kernel є матрицею шару ваг, і bias є вектор зміщення, створений шаром. Найбільш ефективним вирішенням проблеми перенавчання є метод виключення (Dropout).

Побудова моделі виглядає наступним чином (см Рис. 3.13):

```
def build_model():
    l2_val = 0.001
    optimizer_val = 0.001
    dropout_val = 0.5
    model = Sequential()

    model.add(
        Dense(
            128,
            activation="linear",
            input_shape=(len(list(train_data.columns)),),
            kernel_regularizer=keras.regularizers.l2(l2_val),
        )
    )
    model.add(Dropout(dropout_val))
    model.add(
        Dense(128, activation="linear",
              kernel_regularizer=keras.regularizers.l2(l2_val))
    )
    model.add(Dropout(dropout_val))
    model.add(
        Dense(64, activation="linear",
              kernel_regularizer=keras.regularizers.l2(l2_val))
    )
    model.add(Dropout(dropout_val))
    model.add(
        Dense(32, activation="linear",
              kernel_regularizer=keras.regularizers.l2(l2_val))
    )
    model.add(Dense(1))

    optimizer = tf.keras.optimizers.RMSprop(optimizer_val)

    model.compile(loss="hinge", optimizer="adam", metrics=["mae", "mse"])

    return model
```

Рис. 3.13. Побудова моделі

Отримаємо середню абсолютну похибку. У статистиці середня абсолютна похибка (МАП) є мірою помилок між парними спостереженнями, що виражають одне і те ж явище. Приклади Y проти X включають порівняння прогнозованого та спостережуваного, подальшого часу проти початкового часу та одну техніку вимірювання проти альтернативної техніки вимірювання. МАП розраховується як:

$$\text{МАП} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

.

Код обчислення середньї абсолютної похибки (см Рис. 3.14):

```
def plot_history(history):
    hist = pd.DataFrame(history.history)
    hist["epoch"] = history.epoch

    print(hist)

    plt.figure()
    plt.xlabel("Epoch")
    plt.ylabel("Mean Abs Error [MPG]")
    plt.plot(hist["epoch"], hist["mae"], label="Train Error")
    plt.plot(hist["epoch"], hist["val_mae"], label="Val Error")
    plt.legend()
    plt.show()
```

Рис. 3.14. Обчислення середньї абсолютної похибки

Побудуємо графік середньї абсолютної похибки (см Рис. 3.15):

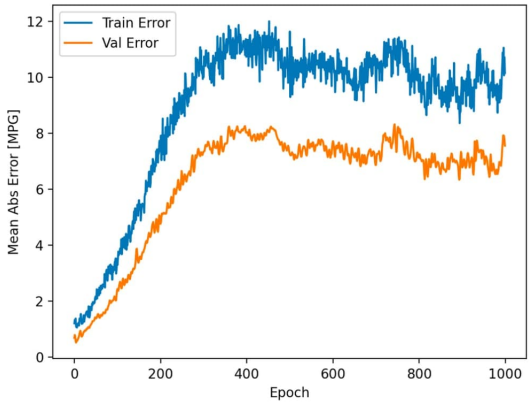


Рис. 3.15. Середня абсолютна похибка

Побудуємо графік помилки передбачення (см Рис. 3.16):

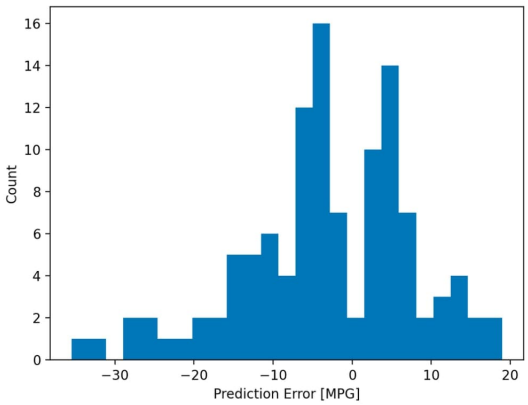


Рис. 3.16. Помилка передбачення

Отримали результат

test_id0 -1.1227461
test_id1 -3.0024393
test_id2 -1.2840757
test_id3 -1.65534
test_id4 -2.3325891
test_id5 1.1599044
test_id6 0.10407401
test_id7 0.24253188

Нормалізуємо

test_id0 0
test_id1 0
test_id2 0
test_id3 0
test_id4 0
test_id5 1
test_id6 1
test_id7 1

Отримали 65% правильно розпізнанних тестових даних, тому що 5 та 6 тестові дані були розпізнанні невірно.

Код обчислення кореляції має вигляд (см Рис. 3.17):

```python
import pandas as pd
import typing

train = pd.read_csv("train.csv").drop("Unnamed: 0", axis=1)
test = pd.read_csv("test.csv").drop("Unnamed: 0", axis=1)

data = train.append(test)
result = data["Давать кредит(число)"].copy()
data = data.drop("Давать кредит(число)", axis=1)

corr_dict: typing.Dict[str, float] = {}

methods = ['pearson', 'kendall', 'spearman']

for method in methods:
    for col in data.columns:
        corr_val = result.corr(data[col], method=method)
        corr_dict[f"{col} ({method})"] = str(corr_val).replace(".", ",")

pd.DataFrame(corr_dict, index=["value"]).T.to_csv("correlation.csv")
```

Рис. 3.17. Обчислення кореляції

Кореляція по критерію Спірмена виглядає наступним чином (см Рис. 3.18):

Колонка и метод	Значение
Сумма кредита (spearman)	-0,473870753059091000
Срок кредита (spearman)	-0,620790892096736000
Дата кредитования12.01.03 (spearman)	-0,096042389495851200
Дата кредитования11.01.03 (spearman)	-0,104438731294419000
Дата кредитования10.01.03 (spearman)	-0,055809447035455000
Дата кредитования09.01.03 (spearman)	-0,055809447035455000
Дата кредитования08.01.03 (spearman)	-0,055809447035455000
Дата кредитования07.01.03 (spearman)	-0,007180162776491280
Дата кредитования06.01.03 (spearman)	0,041449121482472400
Дата кредитования05.01.03 (spearman)	-0,055809447035455000
Дата кредитования04.01.03 (spearman)	0,090078405741436100
Дата кредитования03.01.03 (spearman)	0,138707690000400000
Дата кредитования02.01.03 (spearman)	0,041449121482472400
Дата кредитования01.01.03 (spearman)	0,090078405741436100

Рис. 3.18. Кореляція Спірмена

Обчислимо середнє по модулю значення по трьом критеріям: Пірсона, Кендалла та Спірмена.

Отримали 0.07696393045370722, 0.07531732652416737, 0.0774108554089057.

З метою покращення результатів відкинемо ті дані з датасету, які мали кореляцію нижче середньої по модулю по Спірмену. Найбільш чутливі дані: Сума кредиту, срок кредита, спеціалізація, галузь підприємства (освіта), Дата кредитування 03.01.03, розсташування, немає майна, основний напрямок витрат (купівля товарів довгих. користування)

Як результат отримали

test_id0 1
test_id1 0
test_id2 0
test_id3 0
test_id4 0
test_id5 1
test_id6 1
test_id7 1

тобто 50% правильно розпізнанних даних.

Тоді спробуємо навпаки видалити дані, які мають найбільшу чутливість.

Отримали

test_id0 1
test_id1 1
test_id2 0
test_id3 1
test_id4 0
test_id5 1
test_id6 1
test_id7 1

Тепер результат 37.5%.

ВИСНОВКИ

Для вирішення задачи "можливість видачі кредиту"був використан нейросімулятор з аналітичного пакету "Deductor"[6], а також бібліотека "Tensorflow"та програмна мова "Python"для побудови і використання нейромережі. Навчання нейромережей проходило на базі даних, в якій містились дані про 149 людей. Як результат отримали моделі, з різною чутливістю до різних полей вхідних даних. Для вирішення конкретної задачі видачу кредиту з таким обсягом навчальних даних кращий результат продемонструвала нейромережа. Покращити результат моделі написаної з методом "Deep Learning"можно збільшив набір вхідних даних, більш чутким видаленням вхідних даних, які погано впливають на результат і додавши коефіцієнт конкретним полям задля зменьшення, або збільшення впливу на вихідні дані. На прикладі моделей, побачили що не завжди нейромережі є кращим вибором до розв'язування задач, бо нейромережа може неправильно розпізнати закономірності та приорітизувати одні вхідні дані до інших, як, наприклад, і було продемонстровано.

СПИСОК ЛІТЕРАТУРИ

1. Вербицкий В.В., Реут В.В. Введение в численные методы алгебры. Учебное пособие. – Одесса, ОНУ. – 2015. – 165 с.
2. Осовский С. Нейронные сети для обработки информации. - 2002.
3. Ясницкий Л. Н. Введение в искусственный интеллект. – 2006.
4. Andreas Z., Simulation neuronaler Netze. – 1994.
5. Asuncion A., Newman D. J. UCI Machine Learning Repository. – University of California, Irvine, School of Information and Computer Sciences. – 2007.
6. BaseGroup. Аналітичний пакет "Deductor". – 2014. (https://basegroup.ru/deductor/description)
7. Dreyfus, S. E. Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure. – Journal of Guidance, Control, and Dynamics. – 1990.
8. Oludare I. A. State-of-the-art in artificial neural network applications: A survey. – School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia. – 2018.
9. Rumelhart D.E., Hinton G.E., Williams R.J.. Learning Internal Representations by Error Propagation In: Parallel Distributed Processing. – Cambridge, MA, MIT Press. – 1986
10. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. – 2015.
11. Schmidhuber, J. Deep Learning. – Scholarpedia. – 2015.

Додаток А

Програмна реалізація розробки та використання нейромережі у цілях розв'язку задачі кредитування

```
#init.py
import data_improvements as di
import pandas as pd

data = pd.read_csv(
    "~/Desktop/diploma/CorruptionDetector/data/credit_data.csv", sep=";")
ind = 141

to_classify = ["Пол",
               "Машина", "Класс предприятия",
               "Количество иждевенцев"]

to_zero_or_one = ["Частная собственность", "Квартира", "Загородный дом",
                  "Прописка в данном районе", "Гараж", "Должность",
                  "Гражданское состояние", "Занятость супруга"]


to_drop = ["Давать кредит", "Сумма кредита", "Срок кредита", "Дата кредитования",
           "Цель кредитования", "Образование", "Площадь квартиры", "Способ приобрет
           "Расположение", "Срок эксплуатации машины", "Земельный участок", "Отрасл
           "Специализация", "Среднемес. доход", "Основное направление расходов"]

print(data)

for i in to_drop:
    data = data.drop(i, axis=1)

for i in to_classify:
    print(1, i)
    data = di.classify(data, i)

for i in di.nan_columns(data):
    print(2, i)
    data = di.denanization(data, i)

for i in to_zero_or_one:
    print(3, i)
```

```
    data = di.one_or_null(data, i, data[i].iloc[0]).drop(i, axis=1)

for i in data.columns:
    print(4, i)
    data = di.normalize(data, i)

data = di.compare(data)

train_data_converted = data[:ind]
test_data_converted = data[ind:]

print(train_data_converted)
print(test_data_converted)

train_data_converted.to_csv("train_deleted_high.csv")
test_data_converted.to_csv("test_deleted_high.csv")

from __future__ import absolute_import, division, print_function, unicode_literals

#train_model.py
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout
import tensorflow_probability as tfp

import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
import json

import data_improvements as di

train_data = (
    pd.read_csv("train_deleted_low.csv", index_col=None).dropna(
        axis=1).drop("Unnamed: 0", axis=1)
)

targets = train_data["Давать кредит(число)"].values
```

```
train_data = train_data.drop("Давать кредит(число)", axis=1)

def build_model():
    l2_val = 0.001
    optimizer_val = 0.001
    dropout_val = 0.5
    model = Sequential()

    model.add(
        Dense(
            128,
            activation="linear",
            input_shape=(len(list(train_data.columns)),),
            kernel_regularizer=keras.regularizers.l2(l2_val),
        )
    )
    model.add(Dropout(dropout_val))
    model.add(
        Dense(128, activation="linear",
              kernel_regularizer=keras.regularizers.l2(l2_val))
    )
    model.add(Dropout(dropout_val))
    model.add(
        Dense(64, activation="linear",
              kernel_regularizer=keras.regularizers.l2(l2_val))
    )
    model.add(Dropout(dropout_val))
    model.add(
        Dense(32, activation="linear",
              kernel_regularizer=keras.regularizers.l2(l2_val))
    )
    model.add(Dense(1))

    optimizer = tf.keras.optimizers.RMSprop(optimizer_val)

    model.compile(loss="hinge", optimizer="adam", metrics=["mae", "mse"])

    return model
```

```
def plot_history(history):
    hist = pd.DataFrame(history.history)
    hist["epoch"] = history.epoch

    print(hist)

    plt.figure()
    plt.xlabel("Epoch")
    plt.ylabel("Mean Abs Error [MPG]")
    plt.plot(hist["epoch"], hist["mae"], label="Train Error")
    plt.plot(hist["epoch"], hist["val_mae"], label="Val Error")
    plt.legend()

    plt.show()


def build_value_model():

    early_stop = keras.callbacks.EarlyStopping(
        monitor="val_mean_absolute_error", patience=25
    )

    inputs = train_data
    outputs = targets
    model = build_model()

    print(inputs, str(outputs), len(outputs))

    history = model.fit(
        inputs,
        outputs,
        epochs=1000,
        validation_split=0.2
    )

    plot_history(history)

    test_predictions = np.array(
        model.predict(train_data.iloc[int(len(train_data) * 0.2):])
    )
    test_labels = np.array(targets[int(len(train_data) * 0.2):])
```

```
    error = test_predictions.T[0] - test_labels
    plt.hist(error, bins=25)
    plt.xlabel("Prediction Error [MPG]")
    plt.ylabel("Count")
    plt.show()

    plt.scatter(test_labels, test_predictions)
    plt.xlabel("True Values [MPG]")
    plt.ylabel("Predictions [MPG]")
    plt.axis("equal")
    plt.axis("square")
    plt.xlim([0, plt.xlim()[1]])
    plt.ylim([0, plt.ylim()[1]])
    plt.plot([-10000, 10000], [-10000, 10000])
    plt.show()

    return model


if __name__ == "__main__":
    build_value_model()

#correlation.py
import pandas as pd
import typing

train = pd.read_csv("train.csv").drop("Unnamed: 0", axis=1)
test = pd.read_csv("test.csv").drop("Unnamed: 0", axis=1)

data = train.append(test)
result = data["Давать кредит(число)"].copy()
data = data.drop("Давать кредит(число)", axis=1)

corr_dict: typing.Dict[str, float] = {}

methods = ['pearson', 'kendall', 'spearman']

for method in methods:
    for col in data.columns:
        corr_val = result.corr(data[col], method=method)
```

```
        corr_dict[f"{col} ({method})"] = str(corr_val).replace(".", ",")

pd.DataFrame(corr_dict, index=["value"]).T.to_csv("correlation.csv")

#main.py
import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
from value_model import build_value_model
from data_improvements import normalize

value_model = build_value_model()

test_data = pd.read_csv("test_deleted_low.csv", index_col=None).drop(
    ["Unnamed: 0", "Давать кредит(число)"], axis=1)
predictions = value_model.predict(test_data).T[0]


df = pd.DataFrame(
    {"id": [f"test_id{i}" for i in range(
        len(predictions))], "target": predictions}
)
df = normalize(df, "target")
targ = np.around(df["target"].values, decimals=0)
df["target"] = targ
df["target"].astype(int)
df = df.set_index("id")
df.to_csv("submission_deleted_low.csv", index="id")

print(df)
```