НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
фізико-математичний факультет
кафедра математичного аналізу та теорії ймовірностей

«На правах рукопису»
УДК519.21, 368.91

До захисту допущено:
Завідувач кафедри
_____ Олег КЛЕСОВ
«23» травня 2023 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Страхова та фінансова математика»

зі спеціальності 111 «Математика»

на тему: «Застосування нейронних мереж в актуарних розрахунках»

Виконала:
студентка ІІ курсу, групи ОМ-11мн
Новікова Алла Анатоліївна _____

Керівник:
доктор фізико-математичних наук, професор
Василик Ольга Іванівна _____

Рецензент:
доцент кафедри теорії ймовірностей, статистики
та актуарної математики
Київського національного університету
імені Тараса Шевченка
Яневич Тетяна Олександрівна

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
фізико-математичний факультет
кафедра математичного аналізу та теорії ймовірностей

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність – 111 «Математика»

Освітньо-наукова програма «Страхова та фінансова математика»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олег КЛЕСОВ
«08» лютого 2023 р.

ЗАВДАННЯ
на магістерську дисертацію
Новіковій Аллі Анатоліївні

1. Тема дисертації «Застосування нейронних мереж в актуарних розрахунках», науковий керівник дисертації доктор фізико-математичних наук, професор Василик Ольга Іванівна, затверджені наказом по університету від «27» березня 2023 р. № 1337-с.

2. Термін подання студенткою дисертації «19» травня 2023 року.

3. Об’єкт дослідження: комбінована актуарна нейронна мережа.

4. Предмет дослідження: побудова моделей, визначення страхових тарифів за допомогою моделей комбінованої актуарної нейронної мережі

5. Перелік завдань, які потрібно розробити:

1) ознайомлення з літературою про застосування узагальнених лінійних моделей та нейронних мереж в актуарних розрахунках;

2) вивчення теоретичного обґрунтування побудови комбінованої актуарної нейронної мережі;

3) побудова узагальнених лінійних моделей, нейронних мереж з вбудованими компонентами та комбінованих актуарних нейронних мереж;

4) аналіз отриманих результатів.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу 19 слайдів.

7. Дата видачі завдання «08» лютого 2023 року.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
2.	Опрацювання літератури з тематики магістерської дисертації.	10.02.2023 – 28.02.2023	виконано
3.	Вивчення теоретичного обґрунтування застосування нейронних мереж в актуарних розрахунках	01.03.2023 – 11.03.2023	виконано
4.	Побудова узагальнених лінійних моделей, нейронних мереж з вбудованими компонентами та комбінованих актуарних нейронних мереж в середовищі RStudio.	12.03.2023 – 07.04.2023	виконано
5.	Аналіз результатів моделювання.	07.04.2023 – 21.04.2023	виконано
6.	Написання висновків.	21.04.2023 – 28.04.2023	виконано
7.	Оформлення магістерської дисертації.	29.04.2023 – 18.05.2023	виконано

Студентка Новікова А.А.

Науковий керівник дисертації Василик О.І.

Реферат

Магістерська дисертація: 50 сторінок, 26 першоджерела, 19 слайдів презентації.

Основою забезпечення платоспроможності страхової компанії є актуарні розрахунки. В даній магістерській дисертації досліджується нещодавно запропонована комбінована актуарна нейронна мережа, яка поєднує традиційну узагальнену лінійну модель, що використовується у страховому ціноутворенні, з нейронною мережею. Основна ідея використання нейронної мережі для ціноутворення у страхуванні полягає у моделюванні взаємодії між функціями, які не охоплюються узагальненою лінійною моделлю.

Мета та завдання роботи: застосування знань з теорії ймовірностей, математичної статистики, теорії випадкових процесів, регресійного аналізу, актуарної математики для дослідження нейронних мереж та їх застосування в актуарних розрахунках. Вивчення нового матеріалу про узагальнені лінійні моделі, нейронні мережі, їх комбінації. Самостійною частиною роботи є побудова узагальнених лінійних моделей, нейронних мереж з вбудованими компонентами та комбінованих актуарних нейронних мереж в середовищі RStudio.

Об’єкт дослідження: комбінована актуарна нейронна мережа.

Предмет дослідження: побудова моделей, визначення страхових тарифів за допомогою моделей комбінованої актуарної нейронної мережі.

Ключові слова: страхування, ціноутворення, узагальнена лінійна модель, нейронна мережа, комбінована актуарна нейронна мережа, архітектура, вбудовування шарів, страхування автомобілів, модель регресії Пуассона.

Abstract

Master’s thesis contains 50 pages, 26 primary sources, 19 slides of presentation.

Actuarial calculations are the basis for ensuring the solvency of the insurance company. This master's thesis is devoted to the study of recently proposed combined actuarial neural network that combines the traditional generalized linear model used in insurance pricing with a neural network. The main idea behind using a neural network for insurance pricing is to model interactions between functions that are not covered by a generalized linear model.

The purpose and tasks of the work: application of knowledge of probability theory, mathematical statistics, theory of random processes, regression analysis, actuarial mathematics to the study of neural networks and their application in actuarial calculations. Learning new material about generalized linear models, neural networks, their combinations. An independent part of the work is the construction of generalized linear models, neural networks with built-in components and combined actuarial neural networks in the RStudio environment.

Research object:: combined actuarial neural network.

The subject of research: constructing the models, determining insurance rates using models of a combined actuarial neural network.

Keywords: insurance, pricing, generalized linear model, neural network, combined actuarial neural network, architecture, embedding layers, car insurance, Poisson regression model.

Зміст

Вступ

Страхування базується на принципі, що група людей робить внески до спільного фонду для відшкодування витрат тих, хто постраждав від страхової події. На конкурентному ринку страховики можуть бути прибутковими лише тоді, коли їхні ціни якомога точніше відображають ризики, які вони покривають.

Незважаючи на те, що *узагальнена лінійна модель* (generalized linear model, GLM) є стандартним інструментом ціноутворення у страхуванні, крім страхування життя, починаючи з 1990-х років [10] дослідники та практики постійно прагнуть покращити продуктивність та ефективність процесу моделювання, нещодавно звернувшись до *машинного навчання* (machine learning, ML). Застосування методів машинного навчання в актуарній науці вивчалося і теоретично обґрунтовувалося ([8], [9]), зокрема і застосування *нейронних мереж* (neural networks, NN).

Основними перевагами моделі NN у визначенні страхових тарифів є чудова статистична продуктивність та автоматичне моделювання складних взаємодій між елементами [7]. Перша перевага ще не визначена науково, оскільки в літературі немає єдиної думки щодо неї ([7], [9], [11]), що вказує на те, що результати залежать від ситуації та застосування. Інша перевага є менш неоднозначною при моделюванні за допомогою GLM, оскільки моделювання взаємодій вручну є виснажливим процесом з обмеженою здатністю досліджувати складні взаємозв'язки. Таким чином, нейронні мережі потенційно забезпечують подвійне покращення порівняно з GLM у цій галузі з точки зору часу та продуктивності.

Незважаючи на вищезазначені переваги, використання моделей ML і NN ще не набуло широкого розповсюдження в комерційному страхуванні, головним чином через їхній основний недолік – відсутність інтерпретації. Моделі NN часто називають моделями "чорної скриньки", оскільки їх можна аналізувати лише на вхідних-вихідних даних, а не на внутрішній

роботі. Зокрема, у той час як GLM призначає ваги функціям, вказуючи таким чином на вплив функції на прогнози в простій (зазвичай мультиплікативній) моделі, моделі ML просто створюють прогноз, не пояснюючи, як модель прийшла до такого висновку. Страхові компанії не тільки хочуть розуміти, як їхній ризик розподіляється між клієнтами, але також повинні мати можливість пояснити, що вплинуло на кінцевий розмір страхової премії клієнта. Існують також проблеми впровадження від аналітичного робочого процесу до виробничої системи за відсутності чітко визначеної формули ціноутворення. Ще однією проблемою NN моделей вважається неоднозначність ціноутворення найкращих моделей ML за певного сценарію. На відміну від GLM, дві моделі NN можуть, враховуючи стохастичність процесу навчання, мати однакову статистичну продуктивність, але давати різні індивідуальні прогнози для того самого набору даних [4].

Визначивши переваги та недоліки нейронних мереж, була запропонована нова модель, яка поєднує класичну GLM із нейронною мережею, в якій намагались зберегти переваги обох [12]. Таку модель назвали *комбінованою актуарною нейронною мережею* (CANN). На практиці передбачається, що модель GLM може бути вкладена в NN за допомогою пропуску з'єднання, щоб створити вдосконалення моделі GLM, не відхиляючись надто далеко від свого оригіналу. Це також дозволяє NN досліджувати взаємодію між функціями, які не розглядалися в GLM. Було запропоновано кілька конфігурацій моделі CANN, де розробник моделі може вирішити, чи повинен параметр моделі GLM бути навчальним чи ні в NN [12].

Підхід CANN дозволив покращити статистичні показники GLM-моделей, оскільки відсутні взаємодії можна систематично ідентифікувати [1]. Крім того, є деякі інші переваги запропонованої моделі, а саме автоматизований процес вибору параметрів і швидку збіжність алгоритму

градієнтного спуску, що дозволяє використовувати методи бутстрепінгу (оцінювання на основі інших оцінок) для вивчення точності прогнозу [14].

Підхід CANN зберігає перевагу нейронної мережі в автоматичному визначенні взаємодії складних функцій, а також зменшує «неоднозначність найкращої моделі», оскільки він походить від моделі GLM. Однак основний недолік моделі CANN все ще є: відсутність інтерпретації для NN зберігається в моделі CANN. Таким чином, ключовим питанням для оцінки життєздатності моделей «чорної скриньки» для страхових спеціалістів є краще розуміння того, як їх можна розшифрувати в актуарному контексті. Вплив окремих функцій і взаємодія функцій – це дві ключові сфери, які потребують певного пояснення. В науковій літературі представлено численні методи та інструменти, які спрямовані на інтерпретацію моделей «чорної скриньки» ([15], [16]). Оцінка їх релевантності в актуарних умовах для моделей CANN є важливою частиною оцінки корисності моделей CANN.

Магістерська дисертація складається зі вступу та п’яти розділів. У першому розділі розглядається ціноутворення у страхуванні та термінологія щодо умов страхування і політики.

Другий розділ присвячено узагальненим лінійним моделям, їх визначенні та ідеї, а також застосування в актуарній науці.

У третьому розділі розглядаються нейронні мережі прямого поширення, їх ідея, визначення та структура. Також в цьому розділі наявний алгоритм навчання мережі, регуляризація та скіп-з’єднання.

Четвертий розділ присвячено комбінованим актуарним нейронним мережам. В даному розділі розглядається структура цієї мережі , а також зв’язок між функцією зв’язку та функцією активації виходу і вимірювання похибок.

В п’ятому розділі відбувається дослідження та побудова узагальнених лінійних моделей, нейронних мереж з вбудованими компонентами та комбінованих актуарних нейронних мереж в середовищі RStudio.

1. Ціноутворення у страхуванні

Актуарії та страхові аналітики розробляють моделі ціноутворення для страхових продуктів у процесі, який називається *визначенням страхових тарифів (ставок)*. Визначення ставок передбачає групування ризиків зі схожим потенціалом збитків на основі інформації про страхові поліси та встановлення різних цін, щоб відобразити відмінності в потенціалі збитків між цими групами. Характеристики полісів називаються ознаками, і можуть бути використані розробником ставок як аргументи для моделі ризику. Нижче наведемо деяку важливу термінологію щодо умов страхування та політики.

Означення 1.1. *Експозиція страхового полісу* — це час (у роках), протягом якого поліс був дійсним в межах досліджуваного періоду. *Експозиція для групи страхових полісів* визначається шляхом підсумовування експозицій окремих полісів. *Частота позовів* визначається діленням сумарної кількості позовів на суму експозицій.

Означення 1.2. *Страховий випадок* - подія, передбачена договором страхування або законодавством, ризик виникнення якої застрахований, з настанням якої виникає обов'язок страховика здійснити страхову виплату страхувальнику або іншій особі, визначеній у договорі страхування або відповідно до законодавства [26].

Означення 1.3. *Серйозність позову* — це середній розмір позову, отриманий шляхом ділення сумарної величини позовів на кількість позовів.

Згідно з Законом України про страхування [26], страхові тарифи обчислюються страховиком математичними, статистичними та/або економічними методами з урахуванням статистики настання страхових випадків та ймовірного розміру збитків, характеристик об'єкта страхування, розміру франшизи та інших умов страхування, а за страховими ризиками за класами страхування життя - також з урахуванням величини гарантованого

інвестиційного доходу за цими ризиками, якщо це передбачено договором страхування життя.

Означення 1.4. *Страховий тариф (брутто-тариф)* складається з:

1) *нетто-тарифу*, що включає оцінку страхового ризику, який приймається на страхування за договором страхування, та призначений для формування технічних резервів;

2) *навантаження*, яке включає, зокрема, витрати страховика, пов'язані з укладенням (аквізіційні витрати) та виконанням договору страхування.

Означення 1.5. *Страхова премія* за договором страхування визначається шляхом помноження страхової суми та страхового тарифу (у разі його визначення). Страхова премія за договором страхування, за яким не визначається страховий тариф, розраховується відповідно до умов страхового продукту.

У контексті страхування часто бажано скоригувати загальний рівень прогнозування моделей, щоб він відповідав спостережуваному емпіричному середньому значенню. Цей процес називається *нівеляцією* та здійснюється шляхом коригування прогнозів моделі за допомогою коефіцієнта нівеляції, який є часткою середнього прогнозу моделі для набору даних і середніх емпіричних значень для цього набору даних.

1.1. Ціноутворення у страхуванні, відмінному від страхування життя

Ціноутворення у страхуванні, відмінному від страхування життя, є актуарною сферою, що знаходиться на передовій статистичного моделювання та науки про дані. Це традиційна дисципліна, яка вивчає цикли статистичного моделювання. Як правило, у інших видах страхування, ніж страхування життя, актуарій стикається з проблемою неоднорідного портфеля власників страхових полісів і прагне визначати

ціни з поправкою на ризик для кожного з цих клієнтів. Це класична регресійна задача, яка намагається знайти систематичні ефекти в даних, які характеризують страхувальників. На відміну від багатьох інших областей статистичного моделювання, завдання актуарія полягає не в побудові причинно-наслідкових зв'язків, а, в кращому випадку, в пошуку відповідних проксі, які пояснюють тенденції у виплатах страхових відшкодувань. Актуарії в першу чергу зацікавлені в тому, щоб зробити найкращі можливі прогнози, але на такому рівні складності моделі, який дозволяє їм чітко інформувати керівництво і клієнтів про продукти і ціни [17].

2. Узагальнені лінійні моделі

2.1. Визначення узагальнених лінійних моделей

Узагальнена лінійна модель (generalized linear model, GLM) — це клас регресійних моделей, визначених загальним чином, за допомогою яких можна змоделювати зв'язок між залежною змінною y_i та d незалежними змінними $x_i \in \mathbb{R}^d$ з використанням невідомого параметра $\beta \in \mathbb{R}^d$. Ключовим припущенням GLM є те, що розподіл залежної змінної є членом сімейства експоненційних розподілів. Прикладами є розподіл Гауса, біноміальний, Пуассона, експоненційний і гамма розподіл. Загальний вигляд цих розподілів такий

$$f(y_i, \theta_i, \varphi) = exp\left(\frac{y_i\theta_i - b(\theta_i)}{a(\varphi)} + h(y_i, \varphi)\right), \tag{2.1}$$

φ — параметр масштабу, а θ_i — параметр розташування. Наступні ключові властивості мають місце для членів експоненційного сімейства,

$$\mu = \mathbb{E}[y] = \frac{db(\theta_i)}{d\theta_i}, \tag{2.2}$$

$$Var(y) = \frac{d^2b(\theta_i)}{d\theta_i^2} = \frac{d\mu}{d\theta_i}a(\varphi). \tag{2.3}$$

Основною ідеєю GLM є лінійна модель відповідної функції математичного сподівання залежної змінної. Зазвичай це визначається лінійним предиктором η_i,

$$\eta_i = g(\mathbb{E}[y_i]) = g(\mu_i) = \langle \boldsymbol{x_i}, \boldsymbol{\beta} \rangle. \tag{2.4}$$

Очікувана відповідь або прогноз, таким чином, визначається так:

$$\mathbb{E}[y_i] = g^{-1}(\eta_i) = g^{-1}(\langle \boldsymbol{x_i}, \boldsymbol{\beta} \rangle). \tag{2.5}$$

Функція g називається *функцією зв'язку*, оскільки вона пов'язує лінійний предиктор з очікуваною відповіддю. Таким чином, GLM визначається двома варіантами моделювання: вибором розподілу залежної

змінної та вибором функції зв'язку. Зазвичай функція зв’язку вибирається так, щоб $\eta_i = \theta_i$ для обраної функції розподілу. Методом оцінювання невідомого вектора β є оцінка максимальної правдоподібності (MLE) з алгоритмом, який ґрунтується на ітераційному повторно зваженому методі найменших квадратів [18].

2.2. Узагальнені лінійні моделі в актуарній науці

Існує дві основні причини використання GLM для визначення страхових тарифів. По-перше, GLM працюють із рядом розподілів із сімейства експоненційних, таких як розподіл Пуассона та гамма-розподіл, які, ймовірно, підходять до залежної змінної, що моделюється. По-друге, модель для середнього є не просто лінійною функцією незалежних змінних, як у випадку простої лінійної регресії, а скоріше монотонним перетворенням середнього. Певні розподіли частіше використовуються при визначенні страхових тарифів. Розподіл Пуассона доцільно використовувати для моделювання частоти. Розподіл Твіді рекомендується для моделювання чистої премії, а гамма-розподіл рекомендовано для налаштування серйозності позову тощо [10].

2.3. Від узагальнених лінійних моделей до нейронних мереж

Більшість методів моделювання в актуарному ціноутворенні, які використовуються сьогодні, базуються на фундаментальних роботах [19], [20] про узагальнені лінійні моделі (GLM). Наразі актуарне ціноутворення в автострахуванні зазвичай базується на 40–50 коваріатах, які розрізняють страхувальників. За останні кілька десятиліть актуарії набули великого практичного досвіду в розробці інформації, яка може бути корисною для прогнозного моделювання в GLM. Актуаріям доводиться мати справу з кількома основними проблемами. По-перше, більшість пояснювальних змінних є категоріальними, і, як наслідок, статистичний аналіз стикається зі складними проблемами, наприклад, розрідженістю базової

розрахункової матриці. Крім того, в регресійних функціях коваріати взаємодіють у нетривіальний спосіб, що робить належну оцінку складним завданням. Моделювання частоти страхових випадків — це проблема прогнозування рідкісних подій (проблема дисбалансу класів), де актуарії намагаються знайти систематичні ефекти в даних, у яких значною мірою переважає шум (випадкова частина). Оскільки не існує простої готової моделі розподілу, моделювання розміру страхових виплат має на меті знайти хороший компроміс між складністю та точністю моделі. Це справедливо, наприклад, в сімействі розподілів експоненціальної дисперсії (EDF), які зазвичай не відповідають всьому діапазону розмірів вимог. Це призвело до того, що широко досліджуються все більш складні моделі, що призводить до технічних ускладнень (див., наприклад, [21], [22], [23]).

3. Нейронні мережі

3.1. Ідея нейронних мереж

Для вирішення завдань обробки даних все частіше використовуються методи машинного навчання. Постійно зростаючі бази даних ускладнюють «ручну» розробку моделей, змушуючи актуаріїв все більше покладатися на вивчення таких інструментів, як нейронні мережі. Існують активні спільноти актуаріїв, які стимулюють дослідження в цій галузі, наприклад, ініціатива з науки про актуарні дані Швейцарського інституту актуаріїв.

Нейронна мережа (neural networks, NN) прямого поширення — це модель статистичного навчання, назва якої походить від нейронної структури мозку, яку вона імітує. Метою NN прямого поширення є апроксимація деякої функції $y = f^*(\boldsymbol{x}, \boldsymbol{\theta})$, де вхідний сигнал x використовується для прогнозування вихідного сигналу y. При оцінці параметрів розглядається параметр мережі $\boldsymbol{\theta}$, який дає найкраще наближення функції. Термін «пряме поширення» означає, що інформація передається тільки вперед у мережі, без зворотного зв'язку на виході з будь-якого рівня. В основі NN лежить теорема про універсальне наближення, яка стверджує, що NN із принаймні одним прихованим шаром, в якому задано достатньо прихованих одиниць, може апроксимувати будь-яку неперервну функцію з певними обмеженнями [24].

3.2. Структура нейронної мережі та визначення

Термін «мережа» стосується способу моделювання функції f^* шляхом об'єднання функцій. Наприклад, у мережі, що складається з двох шарів, перший шар можна представити як $f^{(1)}$ і другий як $f^{(2)}$. Тоді складена форма цих шарів буде $f(\boldsymbol{x}) = f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right)$. У NN існує три типи шарів: вхідний шар, приховані шари та вихідний шар. У наведеному

вище прикладі $\boldsymbol{x}$ буде вхідним шаром, $f^{(1)}$ буде прихованим шаром, а $f^{(2)}$ буде вихідним шаром. Таким чином, приховані шари є проміжними між вхідним і вихідним шарами. Кожен шар є вектором із заданою розмірністю, який отримує вхідні дані від попереднього шару, а потім відображає ці значення на наступному шарі. Опис структури NN зазвичай передбачає визначення глибини та ширини моделі; в нашому випадку глибина — це кількість шарів, а ширина вказана для кожного шару як розмірність цього шару.

Вибір глибини та ширини мережі є складною задачею. Згідно з теоремою про універсальне наближення, для представлення будь-якої функції достатньо одного шару. Однак ширина такого шару може бути неймовірно великою і відповідно модель буде складною для навчання. Загалом більш глибокі мережі дають можливість зменшити ширину, необхідну для наближення функції [24]. Знаходження оптимального поєднання глибини та ширини мережі зазвичай досягається шляхом проб і помилок.

Функції, які відображають кожен шар на інший, називаються *прихованими одиницями*, а функція активації, яка відображає останній прихований шар на вихідний, називається *функцією виведення*. Їх може вибрати розробник моделі, і для цього існує декілька варіантів. Враховуючи вхідні дані $\boldsymbol{x}$, пристрій зазвичай спочатку обчислює афінне перетворення $\boldsymbol{z} = \boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b}$, а потім застосовує поелементну нелінійну функцію $\varphi(\boldsymbol{z})$. Тут $\boldsymbol{\omega}, \boldsymbol{b}$ називаються вагами та зміщеннями і, таким чином, представляють параметри мережі. Крім того, $\varphi(\boldsymbol{z})$ називається функцією активації, яка зазвичай відрізняє тип прихованої одиниці. Одиниця для вихідного рівня називається *вихідною одиницею*, і її потрібно вибрати відповідно до типу та розподілу залежної змінної, здебільшого залежно від того, чи є завдання класифікації чи регресії. Трьома найпоширенішими варіантами для прихованих одиниць є випрямлені

лінійні одиниці, логістична сигмоїдна одиниця та гіперболічний тангенс. Важливою особливістю цих одиниць є градієнт, оскільки він використовується для калібрування параметрів мережі [24].

Випрямлені лінійні одиниці (Rectified Linear Units, ReLU) використовують функцію активації $\varphi(z) = max\{0, z\}$. Градієнт ReLU залишається великим, коли блок активний, що сприяє процедурі навчання. Більше того, це дозволяє розріджену активацію, оскільки одиницю можна легко встановити на нуль. Це означає, що мережа, як правило, менш затратна з точки зору обчислень, ніж мережа, яка використовує інші приховані одиниці. Недоліком методу ReLU є те, що він не може навчатися із спостережень, для яких їх активація дорівнює нулю. Загальним підходом для усунення цього недоліку є використання ReLU з витоком, який фіксує константу α до малого значення, наприклад $\alpha = 0.001$, і визначає функцію активації як $\varphi(z) = max\{\alpha z, z\}$.

Сигмоїдна одиниця використовує сигмоїдну функцію як функцію активації, $\varphi(z) = \sigma(z) = \frac{1}{1+e^{-x}}$, а гіперболічний тангенс використовує функцію гіперболічного тангенса $\varphi(z) = th(z)$. Ці дві функції пов'язані $th(z) = 2\sigma(2z) - 1$. Сигмоїдна функція має діапазон $(0,1)$, а функція гіперболічного тангенса – $(-1,1)$. Градієнт цих функцій завжди додатний, близький до лінійного поблизу нуля, але асимптотично спадає. Таким чином, ці одиниці дозволяють прихованим одиницям по суті стати класифікаторами, придатними для певних проблем класифікації. Проте асимптотично спадаючі градієнти створюють проблему при навчанні моделей, оскільки одиниці часто насичуються до низького значення, коли z від'ємне, і до високого значення, коли z додатне [24].

Враховуючи наведені вище визначення, дамо формальне визначення NN прямого поширення. Нехай x_i — вектор вибірок k незалежних змінних для спостереження i, а y_i — вибірка залежної змінної. Розглянемо NN з

$l = 1, \ldots, L$ прихованих шарів, де шар l складається з $q^{(l)}$ прихованих одиниць, позначених $a_q^{(l)}$, де $q = 1, \ldots, q^{(l)}$ і $a^{(l)} = \left\{a_1^{(l)}, \ldots, a_{q^{(l)}}^{(l)}\right\}$. Припустимо, що мережа використовує функцію активації $\varphi(z)$ і афінне перетворення

$$\boldsymbol{z}_q^{(l)} = \langle \boldsymbol{\omega}_q^{(l)}, \boldsymbol{a}^{(l-1)} \rangle + b_q^{(l)}, \tag{3.1}$$

таке, що

$$\boldsymbol{a}_q^{(l)} = \varphi\left(\boldsymbol{z}_q^{(l)}\right) = \varphi\left(\langle \boldsymbol{\omega}_q^{(l)}, \boldsymbol{a}^{(l-1)} \rangle + b_q^{(l)}\right), \tag{3.2}$$

для $q = 1, \ldots, q^{(l)}$, де $\omega_q^{(l)}$ це $q^{(l-1)}$-вимірний вектор ваг від прихованих одиниць у $(l-1)$-му шарі до q-го прихованого блоку в l-му шарі, а $b_q^{(l)}$ — зміщення для q-го прихованого блоку в l-му шарі. Це позначення можна додатково спростити, позначивши

$$\boldsymbol{W}^{(l)} = \left\{\boldsymbol{\omega}_1^{(l)}, \ldots, \boldsymbol{\omega}_{q^{(l)}}^{(l)}\right\} \in \mathbb{R}^{q^{(l-1)} \times q^{(l)}} \tag{3.3}$$

$$\boldsymbol{b}^{(l)} = \left\{b_1^{(l)}, \ldots, b_{q^{(l)}}^{(l)}\right\} \tag{3.4}$$

і нехай $\varphi(z)$ є поелементною функцією, тому

$$\boldsymbol{a}^{(l)} = \varphi\left(W^{(l)T}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}\right). \tag{3.5}$$

Перший рівень прихованих одиниць, $\boldsymbol{a}^{(l)}$ отримує $\boldsymbol{x}_i$ як вхідні дані, а всі наступні рівні $\boldsymbol{a}^{(l)}$ отримують інформацію від $\boldsymbol{a}^{(l-1)}$. Останній шар є просто вихідним рівнем, а також передбаченням $\hat{y}_i = a^{(L+1)}$. На рисунку 1 показано спрощену ілюстрацію NN з виділенням вхідного рівня, прихованих одиниць в одному прихованому шарі та вихідного рівня.

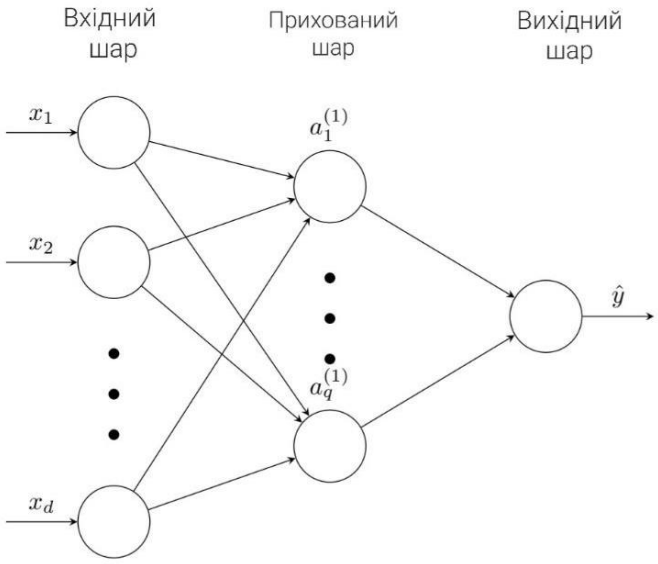

Рис.1: Ілюстрація NN з d входами, прихованим шаром з глибиною q та одним виходом $\hat{y}$.

3.3. Алгоритм навчання мережі

Як і більшість моделей машинного навчання, нейронні мережі використовують градієнтну оптимізацію функції витрат для оцінки параметра мережі $\boldsymbol{\theta} = \{\boldsymbol{W}^{(1)}, \dots, \boldsymbol{W}^{(L+1)}, \boldsymbol{b}^{(1)}, \dots, \boldsymbol{b}^{(L+1)}\}$. Тобто, градієнти функції витрат по відношенню до кожної ваги та зміщення використовуються для оновлення параметра мережі з метою мінімізації функції витрат. Нелінійність нейронних мереж призводить до того, що більшість функцій витрат стають неопуклими, тому алгоритми на основі градієнта лише зменшують функцію витрат до низького значення, але не обов'язково забезпечують збіжність до глобального мінімуму. Існує декілька функцій витрат, і їх придатність залежить від проблеми, що моделюється. Зазвичай в задачах регресії використовується середня квадратична похибка (MSE):

$$C(\hat{y}_i, y_i) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\,, \tag{3.6}$$

де n — кількість значень у вибірці, y_i — спостережуваний результат, а $\hat{y}_i$ — прогнозований результат. Крім того, у випадках, коли припускається,

що залежна змінна походить з даних обчислень, можна використовувати функцію витрат Пуассона [24]:

$$C(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i \log \hat{y}_i). \tag{3.7}$$

Таким чином, у структурі мережі важливим завданням градієнтного навчанні є обчислення всіх градієнтів ваг і зміщень для функції витрат. Це робиться за допомогою методу зворотного поширення, який обчислює градієнти мережі для однієї навчальної точки даних за допомогою ланцюгового правила числення. Для заданої точки вхідних даних спочатку обчислюються значення всіх прихованих одиниць у мережі шляхом поширення вперед у мережі. На вихідному рівні функція витрат обчислюється з використанням спостережуваного та прогнозованого результату мережі. Потім обчислюються градієнти в кожній із прихованих одиниць, використовуючи правило ланцюгових обчислень, зворотного поширення через мережу. У контексті мережі, визначеної в пункті **3.2**, член похибки для кожного прихованого блоку в мережі визначається як часткова похідна функції витрат відносно зваженого входу $z^{(l)}$:

$$\nabla_{z^{(l)}} C = \boldsymbol{\delta}^{(\boldsymbol{l})}. \tag{3.8}$$

Використовуючи ланцюгове правило числення,

$$\boldsymbol{\delta}^{(\boldsymbol{l})} = \nabla_{\boldsymbol{a}^{(l)}} C \odot \varphi'\left(\boldsymbol{z}^{(l)}\right). \tag{3.9}$$

Похибка допускає прості вирази шуканих градієнтів $\nabla_{\boldsymbol{W}^{(l)}} C$ та $\nabla_{\boldsymbol{b}^{(l)}} C$, оскільки за правилом ланцюга

$$\nabla_{\boldsymbol{b}^{(l)}} C = \nabla_{\boldsymbol{z}^{(l)}} C \odot \nabla_{\boldsymbol{b}^{(l)}} z^{(l)} = \boldsymbol{\delta}^{(l)} \tag{3.10}$$

та

$$\nabla_{\boldsymbol{W}^{(l)}} \mathrm{C} = \nabla_{z^{(l)}} C \odot \nabla_{\boldsymbol{W}^{(l)}} z^{(l)} = \boldsymbol{a}^{(l-1)} \odot \boldsymbol{\delta}^{(l)}, \tag{3.11}$$

де рівняння (3.8) було використано разом із визначенням $\boldsymbol{z}^{(l)}$ і, таким чином, часткових похідних відносно $\boldsymbol{W}^{(l)}$ та $\boldsymbol{b}^{(l)}$.

При навчанні мережі та оновленні ваг і зміщень для кожного рівня наближені похідні не застосовуються безпосередньо до попередніх значень, а масштабуються за допомогою параметра швидкості навчання. Параметр швидкості навчання зазвичай вибирається в діапазоні (0, 1], і масштабує довжину кроків, зроблених у напрямку наближення, мінімізуючи функцію витрат. Загалом, велика швидкість навчання зменшує час навчання, але ціною підвищеного ризику перевищення оптимуму. Зазвичай тестують набір параметрів швидкості навчання та аналізують відповідні результати, щоб знайти підходящу швидкість навчання для конкретної моделі та проблеми [24].

Оскільки навчання нейронних мереж часто використовує великі набори даних для якісного узагальнення, то обчислення градієнта для всього набору даних за один прогін є дорогим з обчислювальної точки зору. Стохастичний градієнтний спуск (Stochastic Gradient descend, SGD) використовується майже у всіх версіях нейронних мереж, оскільки він розбиває навчальний набір даних розміром n на m пакетів. Градієнт для кожного пакета обчислюється окремо та застосовується для оновлення параметра мережі $\boldsymbol{\theta}$ для всіх m пакетів. З математичної точки зору, SGD полягає в тому, що градієнт — це математичне сподівання, яке оцінюється на основі даних з цих пакетів. Зазвичай розмір пакету коливається від одного до кількох сотень, незалежно від розміру всього набору даних. Загалом менші розміри пакетів корелюють із кращим узагальненням моделі, можливо, через шум, який вони додають до процесу навчання. Крім того, через високу дисперсію, що виникає при використанні малої кількості даних в пакеті, малі розміри пакетів зазвичай супроводжується невеликою швидкістю навчання для підтримки стабільності навчання. На практиці алгоритм навчання виконується кілька разів для всього набору даних, при цьому кожен запуск називається *епохою*. Таким чином, у кожну епоху набір даних сегментується на випадкові пакети, а потім подається в мережу за допомогою алгоритму SGD [24].

3.4. Регуляризація

Нейронні мережі, як і багато інших алгоритмів машинного навчання, схильні до перенавчання (англ. overfitting). Коли відбувається перенавчання, модель зазвичай працює виключно добре на навчальному наборі даних, але погано для будь-якого тестового набору даних, оскільки в помилці оцінювання домінує дисперсія, а не зміщення [24]. Методи регуляризації відносяться до будь-якого методу, який зменшує помилку тесту, можливо, в обмін на збільшену помилку навчання. Найпоширеніші з них включають регуляризацію L1, регуляризацію L2 та ранню зупинку [24].

3.5. Скіп-з'єднання

Скіп-з'єднання (skip-connection)— це будь-яке з'єднання, яке дозволяє інформації перетікати з рівня t на рівень l, де $t \neq l-1$, оскільки $t = l-1$ визначає звичайний спосіб поширення інформації через мережу. У математичних термінах $\boldsymbol{a}^{(t)}$ надається як вхідні дані для $\boldsymbol{a}^{(l)}$ на додаток до звичайного члена $\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$ через функцію $s\left(\boldsymbol{a}^{(t)}\right)$, так що $\boldsymbol{a}^{(l)} = \varphi\left(\boldsymbol{z}^{(l)} + s\left(\boldsymbol{a}^{(t)}\right)\right)$. Найпоширенішим скіп-з'єднанням, що використовується, є тотожне відображення

$$s\left(\boldsymbol{a}^{(t)}\right) = \langle \mathbb{I}^{q^{(l)}}, \boldsymbol{a}^{(t)} \rangle = \sum_{q=1}^{q(t)} a_q^{(t)}. \tag{3.12}$$

Скіп-з'єднання вперше були запропоновані як метод моделювання нейронних мереж у статті про мережі глибокого навчання для розпізнавання зображень [13], які виявили, що такі з'єднання полегшують оптимізацію більш глибоких нейронних мереж, зберігаючи статистичне підвищення продуктивності від збільшення глибини.

4. Комбінована актуарна нейронна мережа (CANN)

4.1. Структура мережі

Розглянемо GLM із функцією зв'язку, $g(\mathbb{E}[y_i]) = g(\mu_i) = \langle \boldsymbol{x}_i, \boldsymbol{\beta} \rangle$, так що $\mu_i = g^{-1}(\langle \boldsymbol{x}_i, \boldsymbol{\beta} \rangle)$. Модель передбачає розподіл із експоненційного сімейства, властивості якого не важливі на даному етапі. Припустимо, враховуючи цю інформацію, що модель навчена і ефективну оцінку максимальної правдоподібності $\hat{\beta}^{MLE}$ знайдено. Прогноз $\hat{y}_i^{GLM}$ для заданого спостереження i визначається як $\hat{y}_i^{GLM} = g^{-1}(\langle \boldsymbol{x}_i, \boldsymbol{\beta}^{MLE} \rangle)$.

Розглянемо тепер NN прямого поширення з L прихованих одиниць, $l = 0,1, \dots, L, L+1$, де $\boldsymbol{a}^{(l)}$ позначає l-й рівень прихованих одиниць і $\boldsymbol{a}^{(0)} = \boldsymbol{x}_i$, $\boldsymbol{W}^{(l)}$ – матриця ваг $q^{(l-1)} \times q^{(l)}$ і $\boldsymbol{b}^{(l)}$ – вектор зсуву для шару l, а також введемо функції активації φ для прихованих шарів і ψ для вихідного шару. Крім того, нехай між вхідним шаром із ваговою матрицею $\hat{\beta}^{MLE}$ та вихідним шаром існує скіп-з'єднання, такий, що

$$\boldsymbol{a}^{(L+1)} = \psi\big(\boldsymbol{W}^{(L+1)}\boldsymbol{a}^{(L)} + \boldsymbol{b}^{(L+1)} + \langle \boldsymbol{x}_i, \widehat{\boldsymbol{\beta}}^{MLE} \rangle\big). \tag{4.1}$$

Якщо параметр мережі ініціалізовано так, що всі ваги та зміщення дорівнюють 0, а ψ вибрано так, що $\psi(x) = g^{-1}(x)$, тоді

$$\boldsymbol{W}^{(L+1)}\boldsymbol{a}^{(L)} + \boldsymbol{b}^{(L+1)} = 0 \tag{4.2}$$

та

$$\boldsymbol{a}^{(L+1)} = \psi\big(\langle \boldsymbol{x}_i, \widehat{\boldsymbol{\beta}}^{MLE} \rangle\big) = g^{-1}\big(\langle \boldsymbol{x}_i, \widehat{\boldsymbol{\beta}}^{MLE} \rangle\big) = \hat{y}_i^{GLM}. \tag{4.3}$$

Таким чином, мережа ініціалізується для точного прогнозування GLM. Інтуїтивно зрозуміло, що будь-які коригування, які алгоритм навчання вносить до параметра $\theta = \{W^{(1)}, \dots, W^{(L+1)}, b^{(1)}, \dots, b^{(L+1)}\}$, будуть прагнути виправити помилки або залишки GLM.

Що стосується функції витрат, в [1] моделюють частоту претензій і їх GLM передбачає розподіл Пуассона і, отже, використовують

пуассонівську функцію витрат. Немає детального обговорення щодо вибору функції витрат для мережі, але пуассонівська функція витрат також може бути використана для мережі.

4.2. Зв'язок між функцією зв'язку та функцією активації виходу

Ми описали вибір функції зв'язку та відповідної функції активації. У ([12], [1]) обговорюють і використовують лише функцію лог-зв'язку $g(x) = \log(x)$ і відповідно функція активації виходу $\psi(x) = \exp(x)$. Ця конкретна функція активації гарним чином пов'язує GLM і NN, де частина GLM просто приписується фактору з NN. Нехай навчений параметр мережі позначено через $\hat{\theta}$ і для кожного $l = 1, \ldots, L$ нехай $\widehat{\boldsymbol{W}}^{(L)}$ і $\widehat{\boldsymbol{b}}^{(L)}$ будуть навченими вагами та зміщеннями для мережі відповідно. Крім того, нехай $\hat{y}_i^{NN} = \psi\left(\widehat{\boldsymbol{W}}^{(L)}\boldsymbol{a}^{(L-1)} + \boldsymbol{b}^{(L)}\right)$ позначає прогноз NN, а $\hat{y}_i^{CANN}$— прогноз CANN. З експоненціальною функцією активації виходу маємо

$$\hat{y}_i^{CANN} = \exp\left(\widehat{\boldsymbol{W}}^{(L+1)}\boldsymbol{a}^{(L)} + \boldsymbol{b}^{(L+1)} + \langle \boldsymbol{x}_i, \widehat{\boldsymbol{\beta}}^{MLE} \rangle\right) =$$

$$= \exp\left(\widehat{\boldsymbol{W}}^{(L+1)}\boldsymbol{a}^{(L)} + \boldsymbol{b}^{(L+1)}\right) \exp\left(\langle \boldsymbol{x}_i, \widehat{\boldsymbol{\beta}}^{MLE} \rangle\right) = \hat{y}_i^{NN}\hat{y}_i^{GLM}. \qquad (4.4)$$

Результат у рівнянні (4.4) показує, що при використанні функції лог-зв'язку для GLM і експоненціальної функції активації виходу для частини мережі прогноз моделі CANN можна розділити на два множники: коефіцієнт для частини GLM і коефіцієнт для частини NN. Отже, якщо припустити, що GLM робить точні прогнози, то коефіцієнт $\hat{y}_i^{NN}$ буде близьким до 1.

4.3. Вимірювання похибок

У цьому розділі розглядаються три методи вимірювання похибок, щоб оцінити продуктивність моделі: середня квадратична похибка (MSE), середня абсолютна похибка (MAE) і функція витрат Пуассона. Розглянемо

набір даних з n точками даних і з емпіричними чистими преміями $y \in \mathbb{R}^n$, для яких модель зробила прогнози $\hat{y} \in \mathbb{R}^n$. Середня квадратична похибка визначається наступним чином:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 . \tag{4.5}$$

MSE розглядає всі точки даних однаково, що може бути проблематично в ситуаціях, коли деякі точки даних становлять більший інтерес, ніж інші. Щоб переконатися, що квадратичні похибки були зважені відповідно до їх пропорційної ваги в наборі даних, ми визначаємо зважену середню квадратичну похибку (WMSE) як:

$$\text{WMSE} = \frac{1}{\sum_i^n v_i}\sum_{i=1}^{n} v_i(y_i - \hat{y}_i)^2 , \tag{4.6}$$

де v_i — експозиція або вага для точки даних i. MSE є загальним вибором для вимірювання похибки в задачах регресії, оскільки вона зводить похибки в квадрат, що завжди дає додатну похибку. Недоліком MSE є те, що вона дуже чутлива до викидів. Оцінка похибки, яка є більш стійкою до викидів, це середня абсолютна похибка (MAE), яка визначається як:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| . \tag{4.7}$$

Аналогічно до наведеного вище визначення WMSE, зважена середня абсолютна похибка (WMAE) визначається наступним чином:

$$\text{WMAE} = \frac{1}{\sum_i^n v_i}\sum_{i=1}^{n} v_i|y_i - \hat{y}_i| . \tag{4.8}$$

Пуассонівська функція витрат визначається як:

$$\text{Loss} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i \log \hat{y}_i) , \tag{4.9}$$

а також зважені втрати, або WLoss визначається наступним чином:

$$\text{WLoss} = \frac{1}{\sum_i^n v_i} \sum_{i=1}^{n} v_i(\hat{y}_i - y_i \log \hat{y}_i)\,. \qquad (4.10)$$

Витрати Пуассона часто використовуються при моделюванні розподілу Пуассона, наприклад, в узагальнених лінійних моделях, а також при моделюванні нейронних мереж.

5. Застосування нейронних мереж в актуарних розрахунках

Моделювання за допомогою нейронних мереж має недолік, пов’язаний з тим, що не використовує систематичного способу вдосконалення класичних статистичних регресійних моделей. Підхід вкладення класичної GLM в архітектуру NN дозволяє досліджувати структуру моделі, яку не охоплює класична GLM.

В цьому розділі наведено системний підхід до вдосконалення класичних моделей регресії, які використовуються в актуарних розрахунках, за допомогою нейронних мереж. Використовується підхід комбінованої актуарної нейронної мережі (CANN), який пропонує вкладення класичної параметричної регресійної моделі в архітектуру NN для використання обох методів одночасно [1].

Для побудови моделей будемо використовувати французький набір даних страхування цивільно правової відповідальності власників транспортних засобів (ОСЦПВ), який включений в R-пакет «CASdatasets» [25]. Всі обчислення та побудови моделей реалізовано в програмному середовищі RStudio.

5.1. Дані та перегляд узагальнених лінійних моделей

5.1.1. Французькі дані страхування автоцивільної відповідальності

Розглянемо дані «freMTPL2freq», які входять до пакету [25]. Ці дані включають французький портфель страхування ОСЦПВ із відповідною кількістю позовів, спостережуваних протягом одного звітного року.

Маємо 678007 індивідуальних полісів страхування автомобіля (див. рис.2).

```
str(freMTPL2freq)
'data.frame': 678007 obs. of  13 variables:
  $ IDpol     : num  1 3 5 10 11 13 15 17 18 21 ...
$ Exposure  : num  0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
$ Area      : Factor w/ 6 levels "A","B","C","D",..: 4 4 2 2 2 5 5 3 3 2 ...
$ VehPower  : num  5 5 6 7 7 6 6 7 7 7 ...
$ VehAge    : num  0 0 2 0 0 2 2 0 0 0 ...
$ DrivAge   : num  55 55 52 46 46 38 38 33 33 41 ...
$ BonusMalus: num  50 50 50 50 50 50 50 68 68 50 ...
$ VehBrand  : Factor w/ 11 levels "B1","B2","B3",..: 9 9 9 9 9 9 9 9 9 9 ...
$ VehGas    : Factor w/ 2 levels "Diesel","Regular": 2 2 1 1 1 2 2 1 1 1 ...
$ Density   : num  1217 1217 54 76 76 ...
$ Region    : Factor w/ 22 levels "R11","R21","R22",..: 18 18 3 15 15 8 8 20 20 12 ...
$ ClaimTotal: num  0 0 0 0 0 0 0 0 0 0 ...
$ ClaimNb   : num  0 0 0 0 0 0 0 0 0 0 ...
```

Рис. 2: Дані «freMTPL2freq» із R-пакету «CASdatasets».

Для кожного полісу ми маємо 12 змінних:

- IDpol – номер полісу (унікальний ідентифікатор);
- Exposure – загальна експозиція в роках;
- Area – код області (категорійний, порядковий);
- VehPower – потужність автомобіля (категорійна, порядкова);
- VehAge – вік автомобіля в роках;
- DrivAge – вік водія в роках;
- BonusMalus – рівень бонус-малус від 50 до 230 (з опорним рівнем 100);
- VehBrand – марка автомобіля (категорійна, номінальна);
- VehGas – автомобіль на дизелі або звичайному паливі (бінарний);
- Density – щільність населення на км^2 у місті проживання водія;
- Region – регіони у Франції (до 2016 року), вони показані на рис. 3 (категорійний);
- ClaimNb – кількість позовів за даним полісом.

Рис.3: Регіони Франції

5.1.2. Пуассонівське частотне моделювання

Набір даних містить 678007 страхових полісів, для яких ми припускаємо, що кількості позовів N_i від окремих полісів (ClaimNb див. рис. 2) незалежні та розподілені за законом Пуассона з

$$N_i \sim Pois(\lambda(\boldsymbol{x}_i)v_i) \tag{5.1}$$

для заданих обсягів $v_i > 0$ (час експозиції в роках див. рис. 2) і заданої функції частоти позовів $\boldsymbol{x}_i \to (\boldsymbol{x}_i)$, де $\boldsymbol{x}_i$ описує інформацію про особливості полісу i. Усі поліси були активними протягом одного звітного року, і обсяги вважаються пропорційними за часом $v_i \in (0; 1]$ для відповідних часових експозицій.

5.1.3. Попередня обробка функцій для узагальнених лінійних моделей

У Розділі 3 [2] представлений підхід GLM як перша можлива модель регресії для оцінки невідомої функції регресії $\lambda(\cdot)$. У цьому розділі розглянемо вдосконалення цієї GLM.

Потрібно попередньо обробити дані, тому використаємо попередню обробку функцій [2]:

- ✓ Area: вибираємо неперервний (логарифмічний) компонент функції для коду Area, тому ми відображаємо $\{A, \dots, F\} \to \{1, \dots, 6\}$;
- ✓ VehPower: створюємо 6 категорійних класів шляхом злиття груп потужності транспортних засобів, більших і рівних 9;
- ✓ VehAge: ми будуємо 3 категоріальних класи $[0, 1), [1, 10], (10, \infty)$;
- ✓ DrivAge: будуємо 7 категорійних класів $[18, 21), [21, 26), [26, 31), [31, 41), [41, 51), [51, 71), [71, \infty)$;
- ✓ BonusMalus: неперервний логарифмічний компонент функції (обмежуємо значення 150);
- ✓ VehBrand: компонент категоріальної ознаки (всього 11 марок);
- ✓ VehGas: компонент двійкової функції;
- ✓ Density: логарифмічна щільність вибирається як неперервний логарифмічний лінійний компонент;
- ✓ Region: компонент категоріальної ознаки (всього 22 позначки).

Обробка даних в R виглядає наступним чином:

```
###  Функція попередньої обробки для GLM
dat2 <- dat
dat2$AreaGLM <- as.integer(dat2$Area)
dat2$VehPowerGLM <- as.factor(pmin(dat2$VehPower,9))
VehAgeGLM <- cbind(c(0:110), c(1, rep(2,10), rep(3,100)))
dat2$VehAgeGLM <- as.factor(VehAgeGLM[dat2$VehAge+1,2])
dat2[,"VehAgeGLM"] <-relevel(dat2[,"VehAgeGLM"], ref="2")
DrivAgeGLM <- cbind(c(18:100), c(rep(1,21-18), rep(2,26-21), rep(3,31-26),
           rep(4,41-31), rep(5,51-41), rep(6,71-51), rep(7,101-71)))
dat2$DrivAgeGLM <- as.factor(DrivAgeGLM[dat2$DrivAge-17,2])
dat2[,"DrivAgeGLM"] <-relevel(dat2[,"DrivAgeGLM"], ref="5")
dat2$BonusMalusGLM <- as.integer(pmin(dat2$BonusMalus, 150))
dat2$DensityGLM <- as.numeric(log(dat2$Density))
dat2[,"Region"] <-relevel(dat2[,"Region"], ref="R24")
str(dat2)
'data.frame': 678007 obs. of  19 variables:
  $ IDpol        : num  1 3 5 10 11 13 15 17 18 21 ...
 $ Exposure     : num  0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
 $ Area         : Factor w/ 6 levels "A","B","C","D",..: 4 4 2 2 2 5 5 3 3 2 ...
 $ VehPower     : num  5 5 6 7 7 6 6 7 7 7 ...
 $ VehAge       : num  0 0 2 0 0 2 2 0 0 0 ...
 $ DrivAge      : num  55 55 52 46 46 38 38 33 33 41 ...
 $ BonusMalus   : num  50 50 50 50 50 50 50 68 68 50 ...
 $ VehBrand     : Factor w/ 11 levels "B1","B2","B3",..: 9 9 9 9 9 9 9 9 9 9 ...
 $ VehGas       : Factor w/ 2 levels "Diesel","Regular": 2 2 1 1 1 2 2 1 1 1 ...
 $ Density      : num  1217 1217 54 76 76 ...
 $ Region       : Factor w/ 22 levels "R24","R11","R21",..: 18 18 4 15 15 8 8 20 20 12 ...
 $ ClaimTotal   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ ClaimNb      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ AreaGLM      : int  4 4 2 2 2 5 5 3 3 2 ...
 $ VehPowerGLM  : Factor w/ 6 levels "4","5","6","7",..: 2 2 3 4 4 3 3 4 4 4 ...
 $ VehAgeGLM    : Factor w/ 3 levels "2","1","3": 2 2 1 2 2 1 1 2 2 2 ...
 $ DrivAgeGLM   : Factor w/ 7 levels "5","1","2","3",..: 6 6 6 1 1 5 5 5 5 1 ...
 $ BonusMalusGLM: int  50 50 50 50 50 50 50 68 68 50 ...
 $ DensityGLM   : num  7.1 7.1 3.99 4.33 4.33 ...
```

Таким чином, розглядаємо 3 компоненти неперервної функції, 1 компонент бінарної функції і 5 компонентів категорійної функції. VehBrand і Region є категорійними за своєю природою; VehPower, VehAge і DrivAge є неперервними.

Простір ознак X задається наступним чином

$$X \subset [1,6] \times \{0,1\}^5 \times \{0,1\}^2 \times \{0,1\}^6 \times [50,150] \times \{0,1\}^{10} \times \{0,1\} \times [0,11] \times \{0,1\}^{21}. \quad (5.2)$$

Тобто ми маємо $q_0 = 1 + 5 + 2 + 6 + 1 + 10 + 1 + 1 + 21 = 48$ −вимірний простір ознак X, а компоненти ознак у $\{0,1\}^k$ у сумі становлять 0 або 1.

На основі попередньої обробки функцій створюємо першу GLM.

Припущення моделі 5.1 (Модель GLM1) Виберемо простір ознак X, як у (5.2), і визначимо функцію регресії $\lambda: X \to \mathbb{R}_+$ за допомогою

$$x \to \log \lambda(x) = \beta_0 + \sum_{l=1}^{q_0} \beta_l x_l = \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle, \quad (5.3)$$

для вектора параметрів $\boldsymbol{\beta} = \left(\beta_0, \dots, \beta_{q_0}\right)' \in \mathbb{R}^{q_0+1}$. Припустимо для $i \geq 1$

$$N_i \sim Pois(\lambda(\boldsymbol{x}_i) v_i).$$

Розділяємо дані на навчальний набір даних D і тестовий набір даних T. Потім ми підбираємо модель GLM1 з оцінкою максимальної правдоподібності (ОМП) на навчальний набір даних D, мінімізуючи відповідну функцію витрат Пуассона у вибірці

$$\boldsymbol{\beta} \to L(D, \lambda) = \frac{1}{n} \sum_{i=1}^{n} 2N_i \left[\frac{\lambda(\boldsymbol{x}_i) v_i}{N_i} - 1 - \log\left(\frac{\lambda(\boldsymbol{x}_i) v_i}{N_i} \right) \right], \quad (5.4)$$

для β-залежної функції параметричної регресії $\lambda(\cdot) = \lambda_\beta(\cdot)$, і де підсумовування проходить по всіх полісах $1 \leq i \leq n = 610206$ у

наборі даних навчання D. Позначимо результуючу ОМП через $\hat{\beta}$. Це забезпечує оцінку функції регресії $\hat{\lambda}(\cdot) = \lambda_{\hat{\beta}}(\cdot)$.

Результати ОМП моделі GLM1:

```
glm(formula = ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM +
        BonusMalusGLM + VehBrand + VehGas + DensityGLM + Region +
        AreaGLM, family = poisson(), data = learn, offset = log(Exposure))

Coefficients:
  Estimate Std. Error  z value Pr(>|z|)
(Intercept)   -4.4942967  0.0446214 -100.721  < 2e-16 ***
  VehPowerGLM5   0.0607178  0.0229807    2.642 0.008239 **
  ...
  VehAgeGLM3    -0.1963348  0.0154062  -12.744  < 2e-16 ***
  DrivAgeGLM1    0.1025470  0.0470253    2.181 0.029207 *
  ...
  DrivAgeGLM7   -0.0962476  0.0298412   -3.225 0.001258 **
  BonusMalusGLM  0.0272285  0.0003854   70.644  < 2e-16 ***
  VehBrandB2     0.0117071  0.0181737    0.644 0.519459
  ...
  VehBrandB14   -0.1732837  0.0932391   -1.858 0.063100 .
  VehGasRegular -0.1726639  0.0140585  -12.282  < 2e-16 ***
  DensityGLM     0.0452394  0.0149030    3.036 0.002401 **
  RegionR11      0.0117644  0.0291351    0.404 0.686370
  ...
  RegionR94      0.0954215  0.0956822    0.997 0.318631
  AreaGLM        0.0395596  0.0200609    1.972 0.048613 *
  ---
  Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 154156  on 610205  degrees of freedom
Residual deviance: 147295  on 610157  degrees of freedom
```

Якість цієї моделі оцінюється за функцією витрат Пуассона поза вибіркою на тестовому наборі даних T, яка визначається наступним чином

$$L(T,\hat{\lambda}) = \frac{1}{n_T}\sum_{t=1}^{n_T} 2N_t\left[\frac{\lambda(\boldsymbol{x}_t)v_t}{N_t} - 1 - \log\left(\frac{\lambda(\boldsymbol{x}_t)v_t}{N_t}\right)\right], \qquad (5.5)$$

де підсумовування проходить по всіх полісах $1 \le t \le n_T = 67801$ у тестовому наборі даних T .

Результати обчислень наведено в таблиці 1.

	run time	# param.	in-sample loss	out-of-sample loss	average frequency
GLM1	15.6 s	49	24.13863	23.8965	7,36%
GLM2	16 s	48	24.12582	23.88406	7,36%

Таблиця 1: час виконання, кількість параметрів моделі, витрати у вибірці та поза вибіркою (одиниці в 10^{-2}), середня оцінена частота на T.

В [1] показано, як привести DrivAge до неперервної функціональної форми. Тому модифікуємо простір ознак X з (5.2) і функцію регресії λ з (5.3). Замінюємо 7 категорійних вікових класів наступною неперервною функцією

$$\text{DrivAge} \to \beta_l \text{DrivAge} + \beta_{l+1} \log(\text{DrivAge}) + \sum_{j=2}^{4} \beta_{l+j} (\text{DrivAge})^j \ , \qquad (5.6)$$

з параметрами регресії $\beta_l, \ldots, \beta_{l+4}$. Таким чином, ми замінюємо 7 категорійних класів на наведену вище неперервну функціональну форму, що має 5 параметрів регресії. Решта частини функції регресії в (5.3) залишаються незмінними, і ми називаємо цю нову модель моделлю GLM2.

Результати ОМП моделі GLM2:

```
1  glm(formula = ClaimNb ~ VehPowerGLM + VehAgeGLM + BonusMalusGLM +
2         VehBrand + VehGas + DensityGLM + Region + AreaGLM + DrivAge +
3         log(DrivAge) + I(DrivAge^2) + I(DrivAge^3) + I(DrivAge^4),
4       family = poisson(), data = learn, offset = log(Exposure))
5
6  Coefficients:
7                 Estimate Std. Error z value Pr(>|z|)
8    (Intercept)    7.755e+01  5.907e+00  13.128  < 2e-16 ***
9    VehPowerGLM5   6.222e-02  2.298e-02   2.707 0.006781 **
10   ...
11   VehPowerGLM9   2.409e-01  2.507e-02   9.607  < 2e-16 ***
12   VehAgeGLM1    -1.735e-02  3.218e-02  -0.539 0.589845
13   VehAgeGLM3    -2.007e-01  1.541e-02 -13.023  < 2e-16 ***
14   BonusMalusGLM  2.744e-02  3.871e-04  70.891  < 2e-16 ***
15   VehBrandB2     1.281e-02  1.818e-02   0.705 0.480886
16   ...
17   VehBrandB14   -1.733e-01  9.324e-02  -1.859 0.063070 .
18   VehGasRegular -1.721e-01  1.407e-02 -12.229  < 2e-16 ***
19   DensityGLM     4.478e-02  1.490e-02   3.005 0.002657 **
20   RegionR11      1.138e-02  2.914e-02   0.391 0.696165
21   ...
22   RegionR94      9.843e-02  9.568e-02   1.029 0.303611
23   AreaGLM        3.985e-02  2.006e-02   1.987 0.046977 *
24   DrivAge        3.928e+00  3.345e-01  11.744  < 2e-16 ***
25   log(DrivAge)  -4.721e+01  3.586e+00 -13.167  < 2e-16 ***
26   ...
27   I(DrivAge^4)  -1.414e-06  1.918e-07  -7.376 1.63e-13 ***
28   ---
29   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
30
31 (Dispersion parameter for poisson family taken to be 1)
32
33 Null deviance: 154156  on 610205  degrees of freedom
34 Residual deviance: 147217  on 610158  degrees of freedom
35 AIC: 193039
36
37 Number of Fisher Scoring iterations: 6
```

У рядках 1-4 вказана модель GLM2, яка показує конкретну функціональну форму для DrivAge і зберігає незмінними всі інші параметри, а саме дві змінні VehPower і VehAge зберігаються як у моделі GLM1. У рядках 24-27 обчислюється ОМП цієї неперервної реалізації (5.6) функціонального компонента DrivAge.

Результуюча продуктивність поза вибіркою тестових даних T цієї другої моделі, підігнаної до даних навчання D, наведена в таблиці 1. Ми спостерігаємо невелике покращення (поза вибіркою) прогнозної потужності. Віддаємо перевагу цій останній моделі над попередньою. Зауважимо, що ця трансформація зменшила кількість оцінюваних параметрів на 1 з $q_0 + 1 = 49$ до 48.

Неперервна модель GLM2 для віку водія виглядає подібно до категорійного маркування, але вона забезпечує плавний перехід між віковими класами порівняно з моделлю GLM1, і це призводить до значно вищої оцінки для водіїв віком 18-19 років.

Висновок. Було вибрано модель GLM2 як еталонну модель. Ця модель має 48 параметрів для оцінки. Одним із недоліків цієї моделі є те, що вона не досліджує взаємодію між компонентами ознак, окрім множень.

5.2. Вбудовування шарів у нейронні мережі

5.2.1. Визначення нейронної мережі

Підкреслимо два моменти з ([2], [3]):

- q_0 — розмірність простору ознак X із вхідними нейронами $z^{(0)} = \boldsymbol{x} \in X$ (вхідний шар).
- $\varphi : \mathbb{R} \to \mathbb{R}$ є (нелінійною) функцією активації. Далі ми виберемо гіперболічну функцію активації $\varphi(x) = tanh(x)$.

Загальна мережева архітектура з K прихованими шарами для нашої задачі регресії Пуассона отримується шляхом додавання вихідного рівня, як показано нижче

$$\lambda: X \to \mathbb{R}_+, \boldsymbol{x} \to \lambda(\boldsymbol{x}) = \exp\left\{\left\langle \boldsymbol{\omega}^{(K+1)}, \left(z^{(K)} \times \ldots \times z^{(1)}\right)(\boldsymbol{x})\right\rangle\right\}. \qquad (5.7)$$

Тобто ми відображаємо нейрони $z^{(K)}$ останнього прихованого шару на вихідний рівень $\mathbb{R}_+$, використовуючи експоненціальну функцію активації

та вагові коефіцієнти $\boldsymbol{\omega}^{(K+1)}$ (включно з перехопленням). Ця мережева архітектура має глибину K і параметр мережі $\theta \in \mathbb{R}^r$ розмірності $r = \sum_{k=1}^{K+1} q_k(1 + q_{k-1})$, що збирає всі ваги мережі $\boldsymbol{\omega}^{(k)}, k = 1, \ldots, K + 1$, ми встановлюємо $q_{K+1} = 1$. Приклад з $K = 3$ прихованими шарами наведено на рис.5. Мережа має вхідний шар розмірності $q_0 = 9$ і $q_1 = 20, q_2 = 15$ і $q_3 = 10$ прихованих нейронів, що призводить до параметра мережі θ розмірності $r = 686$.

Мережа на рис.4 показує для кожного компонента ознаки одного нейрона у вхідному шарі (синій, зелений і рожевий кольори), таким чином, вхідний рівень розмірності $q_0 = 9$.

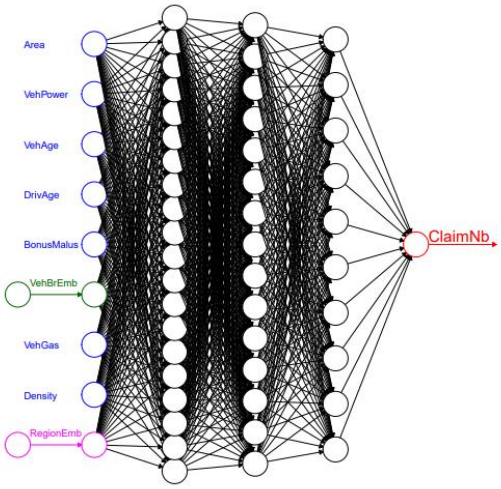

Рис.4: Мережа з $K = 3$ прихованими шарами, що мають $q_1 = 20$, $q_2 = 15$ і $q_3 = 10$ прихованих нейронів у трьох прихованих шарах; вхідний рівень має розміри $q_0 = 9$, що призводить до розмірів мережевих параметрів $r = 686$.

На виході моделі отримуємо прогнозоване значення кількості позовів.

5.2.2. Вбудовування шарів для компонентів категоріальних ознак

Наведемо приклад побудови шару вбудовування на компоненті категоріальної функції VehBrand. Для шару вбудовування нам потрібно вибрати розмір вбудовування $d \in N$. Потім вбудовування визначається наступним чином

$$e: \mathcal{B} \to \mathbb{R}^d, \text{VehBrand} \to e^{\text{VehBrand}} = e(\text{VehBrand}). \quad (5.8)$$

Таким чином, ми призначаємо кожній мітці $\text{VehBrand} \in \mathcal{B}$ $d-$вимірний вектор $e^{\text{VehBrand}} \in \mathbb{R}^d$. Це називається *вбудовуванням* $\mathcal{B}$ у $\mathbb{R}^d$, а ваги вбудовування e^{VehBrand} вивчаються під час калібрування моделі, яке називається *навчанням представлення.*

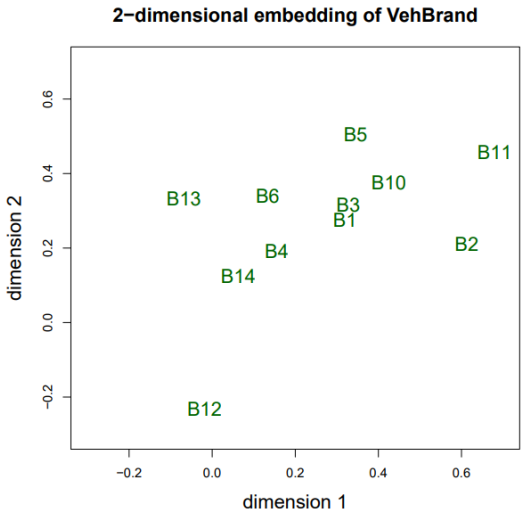

Рис.5: схематична ілюстрація двовимірного вбудовування VehBrand ∈ B.

На рис.5 ілюструємо двовимірне вбудовування $\mathcal{B}$. Це показує для кожної марки транспортного засобу двовимірне представлення, тобто

$$B1 \to e^{B1} \in \mathbb{R}^2,$$

$$B4 \to e^{B4} \in \mathbb{R}^2,$$

...

$$B12 \to e^{B12} \in \mathbb{R}^2.$$

Схематична ілюстрація на рис.5 має наступну інтерпретацію. Транспортний засіб марки $B12$ значно відрізняється від усіх інших марок транспортних засобів, а марки транспортних засобів $B1$ і $B3$ мають подібність, проілюстровану невеликою евклідовою відстанню між цими двома марками транспортних засобів. Якщо ми вбудовуємо два компоненти категоріальних ознак VehBrand і Region у рівні вбудовування розмірності $d = 1$ кожен, тоді ці вбудовування використовують 11 + 22 =

33 вагові коефіцієнти вбудовування. Для вкладення розмірності $d = 2$ кожна, маємо $11 \cdot 2 + 22 \cdot 2 = 66$ ваг вкладень.

5.2.3. Приклад шару вбудовування

Вибираємо мережі глибиною $K = 3$, що мають приховані нейрони $(q_1, q_2, q_3) = (20, 15, 10)$, $(q_1, q_2, q_3) = (25, 20, 15)$ та $(q_1, q_2, q_3) = (10, 10, 15)$.

Побудова мережевої архітектури з шарами вбудовування для категоріальних пояснювальних змінних VehBrand і Region має наступний вигляд $d = 1$:

```
### Нейронна мережа з вбудованими компонентами (EmbNN)
EmbNN <- 0

if (EmbNN==0){
  Vlearn <- as.matrix(log(learn$Exposure))
  Vtest <- as.matrix(log(test$Exposure))
  lambda.hom <- sum(learn$ClaimNb)/sum(learn$Exposure)
}
lambda.hom

d <- 1
# Визначення архітектуру мережі
Design   <- layer_input(shape = c(7),  dtype = 'float32', name = 'Design')
VehBrand <- layer_input(shape = c(1),   dtype = 'int32', name = 'VehBrand')
Region   <- layer_input(shape = c(1),   dtype = 'int32', name = 'Region')
LogVol   <- layer_input(shape = c(1),   dtype = 'float32', name = 'LogVol')
#
BrandEmb = VehBrand %>%
  layer_embedding(input_dim = 11, output_dim = d, input_length = 1, name = 'BrandEmb') %>%
  layer_flatten(name='Brand_flat')

RegionEmb = Region %>%
  layer_embedding(input_dim = 22, output_dim = d, input_length = 1, name = 'RegionEmb') %>%
  layer_flatten(name='Region_flat')

Network = list(Design, BrandEmb, RegionEmb) %>% layer_concatenate(name='concate') %>%
  layer_dense(units=20, activation='tanh', name='hidden1') %>%
  layer_dense(units=15, activation='tanh', name='hidden2') %>%
  layer_dense(units=10, activation='tanh', name='hidden3') %>%
  layer_dense(units=1, activation='linear', name='Network',
              weights=list(array(0, dim=c(10,1)), array(log(lambda.hom), dim=c(1))))

Response = LogVol %>% layer_add(Network) %>%
  layer_dense(units=1, activation=k_exp, name = 'Response', trainable=FALSE,
              weights=list(array(1, dim=c(1,1)), array(0, dim=c(1))))

model.nn <- keras_model(inputs = c(Design, VehBrand, Region, LogVol), outputs = c(Response))
model.nn %>% compile(optimizer = optimizer_nadam(), loss = 'poisson')
```

```
# Підгонка нейронної мережі
{t1 <- proc.time()
fit <- model.nn %>% fit(list(Xlearn, Brlearn, Relearn, Vlearn), Ylearn, epochs=100,
                        batch_size=10000, verbose=0, validation_split=0)
(proc.time()-t1)}
t1[1:5]
plot(fit)

# Розрахунок прогнозів
learn0$fitNN <- as.vector(model.nn %>% predict(list(Xlearn, Brlearn, Relearn, Vlearn)))
test0$fitNN <- as.vector(model.nn %>% predict(list(Xtest, Brtest, Retest, Vtest)))

# Втрати у вибірці та поза вибіркою (у 10^(-2))
Poisson.Deviance(learn0$fitNN, as.vector(unlist(learn0$ClaimNb)))
Poisson.Deviance(test0$fitNN, as.vector(unlist(test0$ClaimNb)))
sum(test0$fitNN)/sum(test0$Exposure)
```

Результати, отримані за допомогою цих моделей з $d = 1$ і $d = 2$, наведено в таблиці 2. Моделі з $d = 2$ мають більше параметрів, що також забезпечує більше ступенів свободи для методу градієнтного спуску.

Запускаємо алгоритм градієнтного спуску для 100 епох на наборі даних навчання D на міні-групах розміром 10 000 полісів. Щоб відстежити підгонку, розділено навчальні дані у співвідношенні 9:1 на набір навчальних даних і набір даних перевірки.

	run time	# param.	in-sample loss	out-of-sample loss	average frequency
GLM2	16 s	48	24.12582	23.88406	7,36%
Network Emb1 (d=1)	22.9 s	719	23.97642	23.77052	7,763153%
Network Emb1 (d=2)	24.93 s	858	23.94253	23.74841	7,578042%
Network Emb2 (d=1)	23.31 s	1134	24.02495	23.84488	8,321029%
Network Emb2 (d=2)	24.37 s	1283	23.99005	23.8153	8,154033%
Network Emb3 (d=1)	22.60s	424	24.0955	23.89068	7,540553%
Network Emb3 (d=2)	23.92 s	543	24.03243	23.7998	7,695127%

Таблиця 2: час роботи, кількість параметрів моделі, витрати у вибірці та поза вибіркою (одиниці в 10^{-2}) моделей GLM2, шарами вбудовування з розмірами $d = 1, 2$ із наборами нейронів $(q_1, q_2, q_3) = (20, 15, 10)$,

$(q_1, q_2, q_3) = (25, 20, 15)$ та $(q_1, q_2, q_3) = (10, 10, 15)$, для компонентів категоріальних ознак VehBrand і Region.

Зазначимо, що отримуємо явно кращу модель, ніж модель GLM2, з точки зору функції витрат Пуассона, проте ціною більшого часу роботи. Найменшу витрату у вибірці та поза вибіркою можна побачити для моделі Network Emb1 (d=2) із $(q_1, q_2, q_3) = (20, 15, 10)$ нейронами. Модель Network Emb2 (d=2) із $(q_1, q_2, q_3) = (25, 20, 15)$ має більше параметрів, що забезпечує більше ступенів свободи для методу градієнтного спуску.

Підігнані моделі з шарами вбудовування явно перевершують GLM2 з точки зору витрат поза вибіркою (див. рис. 6). Калібрування мережевих моделей нестабільне, що призводить до коливання середніх частот, див. останній стовпець у таблиці 2. Насправді ці цифри (не будучи частиною цільової функції під час калібрування моделі) досить сильно коливаються, що є основною проблемою для ціноутворення у страхуванні.

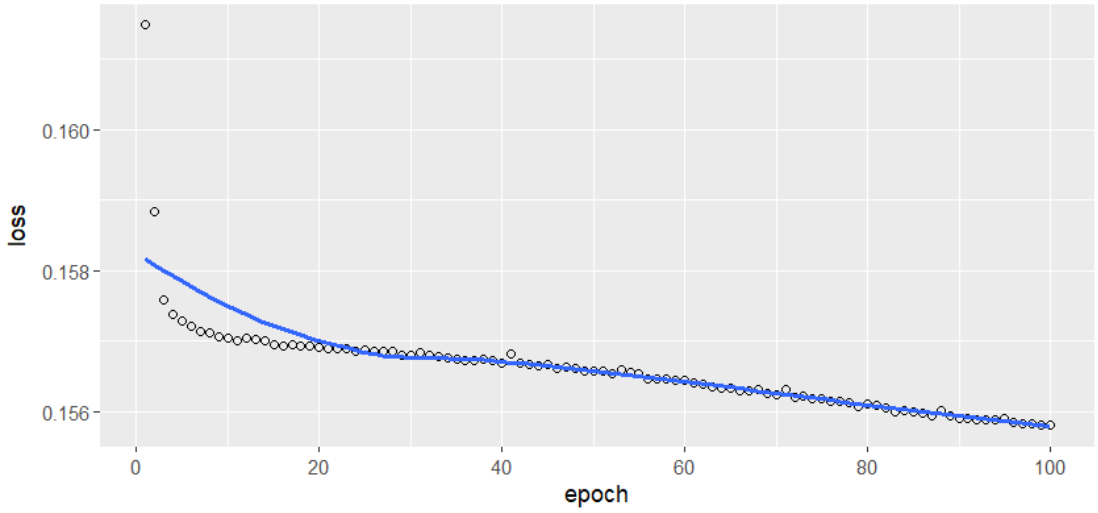

Рис.6: Підгонка нейронної мережі Network Emb1 (d=2) із $(q_1, q_2, q_3) = (20, 15, 10)$ прихованими нейронами.

Висновки. Нейронні мережі покращують результати GLM з точки зору витрат за межами вибірки, оскільки ми не докладаємо достатніх зусиль для пошуку оптимального GLM щодо розробки функцій і потенційних взаємодій. Окрім того, що вбудовування шарів може покращити продуктивність мережі поза вибіркою, вони дозволяють нам візуально ідентифікувати зв'язки між різними рівнями категоріальних вхідних

даних. Недоліками мереж є те, що калібрування призводить до мінливих оцінок середньої частоти та коливань зміщення.

5.3. Комбінований підхід актуарної нейронної мережі

5.3.1. Вкладення актуарної моделі в мережеву архітектуру

У цьому розділі поєднуємо класичну узагальнену лінійну модель і нейронну мережу. Ідея полягає в тому, щоб вкласти GLM в мережеву архітектуру.

Припущення моделі 5.2 (підхід CANN: частина I). Виберемо простір ознак $X \subset \mathbb{R}^{q_0}$ і визначимо функцію регресії $\lambda: X \to \mathbb{R}_+$ за допомогою

$$\boldsymbol{x} \to \log \lambda(\boldsymbol{x}) = \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle + \langle \boldsymbol{\omega}^{(K+1)}, \left(z^{(K)} \times \ldots \times z^{(1)}\right)(\boldsymbol{x}) \rangle, \qquad (5.9)$$

де перший член у правій частині (5.9) є функцією регресії з припущень моделі 5.1 з вектором параметрів $\boldsymbol{\beta}$, а другий член є функцією регресії з (5.7) з параметром мережі θ. Припустимо $N_i \sim Pois(\lambda(\boldsymbol{x}_i)v_i)$ для всіх $i \geq 1$.

Підхід CANN із припущеннями моделі 5.2 проілюстровано на рис.7. Пропускне з'єднання помаранчевого кольору містить GLM (зауважимо, що на даний момент ми нехтуємо тим, що компоненти категорійних функцій можуть використовувати інше кодування для GLM і мережевих частин).

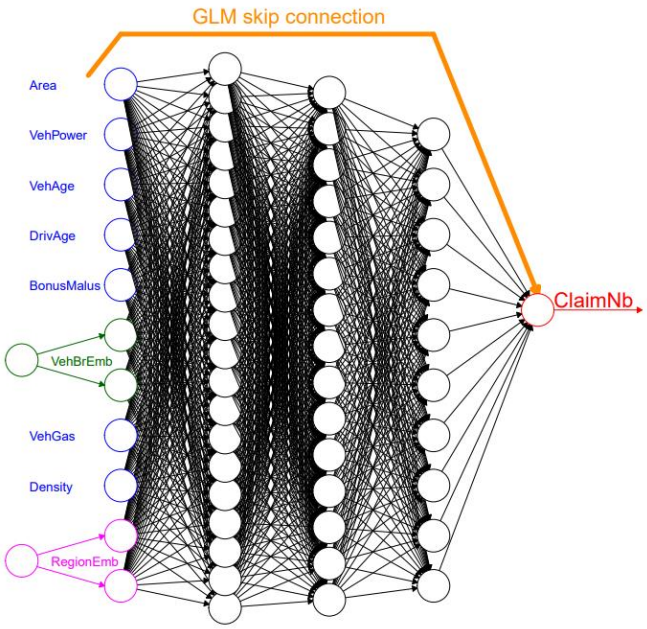

Рис.7: Підхід CANN, який ілюструє помаранчевим кольором класичний GLM зі скіп-з'єднанням, доданим до мережі глибиною $K = 3$ із $(q_1, q_2, q_3) = (20, 15, 10)$ прихованими нейронами.

Зауваження:

- Формула (5.9) поєднує наші попередні дві моделі, зокрема, вона вбудовує GLM в мережеву архітектуру, вбудовуючи її в так зване скіп-з'єднання, яке безпосередньо пов'язує вхідний рівень із вихідним рівнем, див. оранжеву стрілку на рис.7. Скіп-з'єднання використовуються в глибоких мережах, оскільки вони мають хороші властивості калібрування, потенційно уникаючи проблеми зникнення градієнта [13].
- Дві моделі поєднуються у вихідному шарі шляхом (простого) додавання. Це додавання робить один із перетинів β_0 і $\omega_0^{(K+1)}$ зайвим. Тому фіксуємо одне з перехоплень, у більшості випадків β_0, і тренуємо лише інший перетин, скажімо, $\omega_0^{(K+1)}$ у новий параметр мережі $\vartheta = (\boldsymbol{\beta}, \theta)$ функції регресії (5.9).
- Функція регресії (5.9) вимагає, щоб GLM і мережева модель були визначені в одному просторі X. Це може вимагати об'єднання простору функцій моделі GLM і мережевий підхід, а не обидві

частини функції регресії (5.9) можуть враховувати всі компоненти цього об'єднаного простору функцій.

Другим важливим моментом є наступна ідея.

Ініціалізація 5.3 (підхід CANN: частина II) Нехай припущення моделі 5.2 виконуються і що $\hat{\beta}$ позначає ОМП для β згідно з припущеннями моделі 5.1. Ініціалізуємо функцію регресії (5.9) наступним чином: задамо для параметра мережі $\vartheta = (\boldsymbol{\beta}, \theta)$ початкове значення

$$\vartheta_0 = \left(\widehat{\boldsymbol{\beta}},\ \theta_0\right) \textit{ з вагою вихідного шару } \boldsymbol{\omega}^{(K+1)} \equiv 0 \textit{ в } \theta_0. \qquad (5.10)$$

Зауважимо, що ініціалізація (5.10) точно забезпечує передбачення ОМП частини GLM моделі CANN (5.9), тобто вона мінімізує функцію витрат Пуассона згідно з припущеннями моделі 5.1. Якщо ми запускаємо алгоритм градієнтного спуску для підгонки моделі CANN (5.9) до цього початкового значення ϑ_0, і якщо ми використовуємо функцію витрат Пуассона як цільову функцію, тоді алгоритм досліджує архітектуру мережі для додаткової структури моделі, якої немає в GLM і який знижує початкові витрати пуассонівського відхилення, пов'язані з (початковим) параметром мережі ϑ_0. Таким чином, отримуємо вдосконалення GLM за мережевими можливостями. Це забезпечує більш систематичний спосіб використання мережевих архітектур для покращення GLM.

5.3.2. Варіанти підходу CANN

Розглянемо деякі варіанти підходу CANN (5.9)-(5.10). Підхід CANN запускає алгоритм градієнтного спуску в регресійній моделі

$$\boldsymbol{x} \to \lambda(\boldsymbol{x}) = \exp\left\{\langle\widehat{\boldsymbol{\beta}}, \boldsymbol{x}\rangle + \langle\boldsymbol{\omega}^{(K+1)}, \left(z^{(K)} \times \ldots \times z^{(1)}\right)(\boldsymbol{x})\rangle\right\}, \qquad (5.11)$$

де $\widehat{\boldsymbol{\beta}}$ є ОМП для β. Існує два різні способи застосування алгоритму градієнтного спуску до (5.11):

- навчається весь параметр мережі $\vartheta = (\beta, \theta)$,
- оголошується частина GLM $\widehat{\boldsymbol{\beta}}$ непридатною для навчання та навчаємо другий доданок у $\vartheta = (\widehat{\boldsymbol{\beta}}, \theta)$.

В останньому випадку оптимальний GLM завжди залишається у функції регресії CANN і модифікується мережевою частиною.

У першому випадку оптимальний GLM модифікується взаємодією з мережевою частиною.

Варіант (5.11) у випадку, коли ми оголошуємо $\widehat{\boldsymbol{\beta}}$ таким, що не піддається навчанню, полягає у введенні вагового коефіцієнта, що піддається навчанню $\alpha \in [0,1]$, і визначаємо нову регресійну модель

$$\boldsymbol{x} \to \lambda(\boldsymbol{x}) = \exp\{\alpha\langle\widehat{\boldsymbol{\beta}}, \boldsymbol{x}\rangle + (1-\alpha)\langle\boldsymbol{\omega}^{(K+1)}, \left(z^{(K)} \times \ldots \times z^{(1)}\right)(\boldsymbol{x})\rangle\}, \quad (5.12)$$

Якщо ми навчимо цю модель, то дізнаємося вагу достовірності α, за якої GLM розглядається в підході CANN.

5.3.3. Приклад та загальна реалізація CANN

Розглянемо приклад, який реалізує підхід CANN (5.11), де ми оголошуємо ОМП $\widehat{\boldsymbol{\beta}}$ моделі GLM2 такою, що не піддається навчанню, і використовуємо $d = 1,2$ для рівнів вбудовування частини мережі із наборами прихованих нейронів $(q_1, q_2, q_3) = (20, 15, 10)$, $(q_1, q_2, q_3) = (25, 20, 15)$ та $(q_1, q_2, q_3) = (10, 10, 15)$.

У випадку розподілу Пуассона ми можемо істотно спростити реалізацію CANN. З (5.11) ми бачимо, що якщо частина GLM не піддається навчанню з MLE $\widehat{\boldsymbol{\beta}}$, тоді ми можемо об'єднати цей член із заданими обсягами v_i. Таким чином, розглядаємо мережеву функцію

$$\boldsymbol{x} \to \log \lambda^{NN}(\boldsymbol{x}) = \langle\boldsymbol{\omega}^{(K+1)}, \left(z^{(K)} \times \ldots \times z^{(1)}\right)(\boldsymbol{x})\rangle, \quad (5.13)$$

і припускаємо, що

$$N_i \sim Pois\left(\lambda^{NN}(\boldsymbol{x}_i) v_i^{GLM}\right) \textit{ з робочими вагами } v_i^{GLM} = v_i \exp\langle\widehat{\boldsymbol{\beta}}, \boldsymbol{x}\rangle. \quad (5.14)$$

Це призводить до простішого представлення моделі CANN.

Побудова моделі CANN з $d = 1$ та $(q_1, q_2, q_3) = (20, 15, 10)$:

```r
### Комбінована актуарна нейронна мережа (CANN)
CANN<-1
if (CANN==1){
  Vlearn <- as.matrix(log(learn$fitGLM2))
  Vtest <- as.matrix(log(test$fitGLM2))
  lambda.hom <- sum(learn$ClaimNb)/sum(learn$fitGLM2)
}
lambda.hom

d <- 1        # розміри, що вбудовують шари для категоріальних функцій
# Визначення архітектуру мережі
Design   <- layer_input(shape = c(7),  dtype = 'float32', name = 'Design')
VehBrand <- layer_input(shape = c(1),   dtype = 'int32', name = 'VehBrand')
Region   <- layer_input(shape = c(1),   dtype = 'int32', name = 'Region')
LogVol   <- layer_input(shape = c(1),   dtype = 'float32', name = 'LogVol')
#
BrandEmb = VehBrand %>%
  layer_embedding(input_dim = 11, output_dim = d, input_length = 1, name = 'BrandEmb') %>%
  layer_flatten(name='Brand_flat')

RegionEmb = Region %>%
  layer_embedding(input_dim = 22, output_dim = d, input_length = 1, name = 'RegionEmb') %>%
  layer_flatten(name='Region_flat')

Network = list(Design, BrandEmb, RegionEmb) %>% layer_concatenate(name='concate') %>%
  layer_dense(units=20, activation='tanh', name='hidden1') %>%
  layer_dense(units=15, activation='tanh', name='hidden2') %>%
  layer_dense(units=10, activation='tanh', name='hidden3') %>%
  layer_dense(units=1, activation='linear', name='Network',
              weights=list(array(0, dim=c(10,1)), array(log(lambda.hom), dim=c(1))))

Response = LogVol %>% layer_add(Network) %>%
  layer_dense(units=1, activation=k_exp, name = 'Response', trainable=FALSE,
              weights=list(array(1, dim=c(1,1)), array(0, dim=c(1))))

model.cann <- keras_model(inputs = c(Design, VehBrand, Region, LogVol), outputs = c(Response))
model.cann %>% compile(optimizer = optimizer_nadam(), loss = 'poisson')
```

```r
# Підгонка нейронної мережі
{t1 <- proc.time()
  fit <- model.cann %>% fit(list(Xlearn, Brlearn, Relearn, Vlearn), Ylearn, epochs=100,
                            batch_size=10000, verbose=0, validation_split=0)
  (proc.time()-t1)}
t1[1:5]
plot(fit)

# Розрахунок прогнозів
learn0$fitCANN <- as.vector(model.cann %>% predict(list(Xlearn, Brlearn, Relearn, Vlearn)))
test0$fitCANN <- as.vector(model.cann %>% predict(list(Xtest, Brtest, Retest, Vtest)))

# Втрати у вибірці та поза вибіркою (у 10^(-2))
Poisson.Deviance(learn0$fitCANN, as.vector(unlist(learn0$ClaimNb)))
Poisson.Deviance(test0$fitCANN, as.vector(unlist(test0$ClaimNb)))
sum(test0$fitCANN)/sum(test0$Exposure)
```

	run time, s	# param.	in-sample loss	out-of-sample loss	average frequency
GLM2	16	48	24.12582	23.88406	7,36%
CANN1 (d=1)	27.43	719	23.69886	23.56614	7,14883
CANN1 (d=2)	29.12	858	23.64881	23.5292	7,137131%
CANN2 (d=1)	28.03	1134	23.68078	23.54529	6,95358%
CANN2 (d=2)	29.59	1283	23.71245	23.57725	6,712538%
CANN3 (d=1)	25.37	424	23.77527	23.61214	7,683855%
CANN3 (d=2)	27,56	543	23.72294	23.58072	7,164337%

Таблиця 3: час роботи, кількість параметрів моделі, витрати у вибірці та поза вибіркою (одиниці в 10^{-2}) моделей GLM2, CANN з вбудовуваннями d = 1, 2 та наборами прихованих нейронів $(q_1, q_2, q_3) = (20, 15, 10)$, $(q_1, q_2, q_3) = (25, 20, 15)$ та $(q_1, q_2, q_3) = (10, 10, 15)$.

Бачимо, що підхід CANN призводить до покращення в порівнянні з класичним мережевим підходом щодо втрат за межами вибірки.

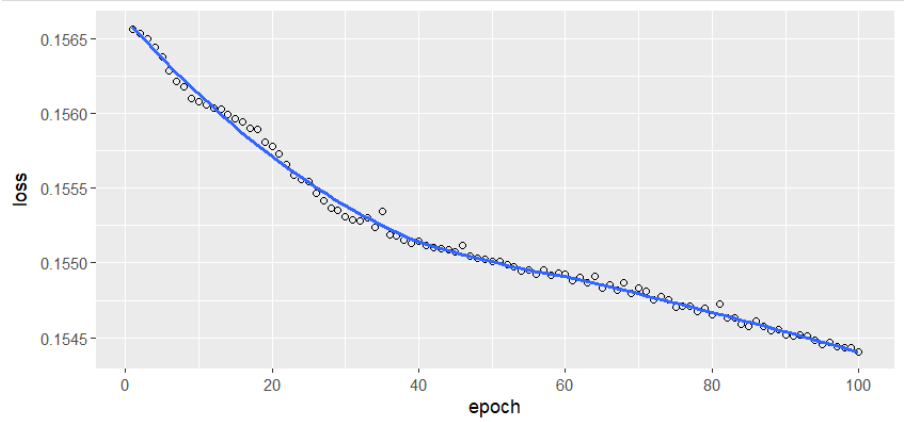

Рис.8: Підгонка нейронної мережі CANN1 (d=2) із $(q_1, q_2, q_3) = (20, 15, 10)$ прихованими нейронами.

Висновок. Граничне моделювання, що використовується в моделі GLM2, можна трохи покращити, але воно не пояснює великих відмінностей між моделлю GLM2 і моделями нейронної мережі в таблиці 3. Таким чином, основний недолік моделі GLM2 порівняно з моделями нейронної мережі має полягати у відсутності взаємодій у попередній моделі.

Висновки

У цій магістерській дисертації розглядаються узагальнені лінійні моделі та нейронні мережі з точки зору їх застосування у страхуванні.

А саме, наведено підхід до вдосконалення класичних моделей регресії, які використовуються в актуарних розрахунках, за допомогою нейронних мереж. Використовується модель комбінованої актуарної нейронної мережі, яка передбачає вкладення класичної параметричної регресійної моделі в архітектуру нейронної мережі для використання обох методів одночасно.

Самостійна частина магістерської дисертації полягає у побудові узагальнених лінійних моделей, нейронних мереж з вбудованими компонентами та комбінованих актуарних нейронних мереж в середовищі Rstudio. Також виконано порівняльний аналіз цих моделей.

В результаті дослідження встановлено, що з точки зору похибки прогнозу використання комбінованої актуарної нейронної мережі призводить до покращення моделі в порівнянні з узагальненими лінійними моделями та нейронними мережами.

Список використаної літератури

1. Jürg Schelldorfer and Mario V. Wüthrich. Nesting classical actuarial models into neural networks. 2019.
ULR https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3320525
2. Noll, A., Salzmann, R., W¨uthrich, M.V. (2018). Case study: French motor third-party liability claims. SSRN Manuscript ID 3164764. Version November 8, 2018. ULR https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3164764
3. Ferrario, A., Noll, A., W¨uthrich, M.V. (2018). Insights from inside neural networks. SSRN Manuscript ID 3226852. Version November 14, 2018.
ULR https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3226852
4. Axel Gustafsson, Jacob Hansen. Combined Actuarial Neural Networks in Actuarial Rate Making.
ULR https://kth.diva-portal.org/smash/get/diva2:1656340/FULLTEXT01.pdf
5. R code. ULR
https://github.com/JSchelldorfer/ActuarialDataScience/tree/master/3%20-%20Nesting%20Classical%20Actuarial%20Models%20into%20Neural%20Networks
6. Daniel Meier and Jürg Schelldorfer, with support from Christian Lorentzen, Friedrich Loser, Michael Mayer, Mario V. Wüthrich and Mirai Solutions GmbH. French Motor Third-Party Liability Claims. ULR
https://htmlpreview.github.io/?https://github.com/JSchelldorfer/ActuarialDataScience/blob/master/3%20-%20Nesting%20Classical%20Actuarial%20Models%20into%20Neural%20Networks/freMTPLfreq_cann.html
7. Charles Dugas, Y. Bengio, Nicolas Chapados, P. Vincent, G. Denoncourt, and C. Fournier.
Statistical learning algorithms applied to automobile insurance ratemaking. 12 2003. doi: 10.1142/9789812794246_0004.
8. Ronald Richman. Ai in actuarial science – a review of recent advances – part 1. Annals of Actuarial

Science, page 1–23, 2020. doi: 10.1017/S1748499520000238.
9. Christopher Blier-Wong, Hélène Cossette, Luc Lamontagne, and Etienne Marceau. Machine learning in pamp;c insurance: A review for pricing and reserving. Risks, 9(1), 2021. ISSN 2227-9091. doi: 10.3390/risks9010004. URL https://www.mdpi.com/2227-9091/9/1/4.
10. Esbjörn Ohlsson and Björn Johansson. Non-Life Insurance Pricing with Generalized Linear Models. EAA Lecture Notes. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 3642107907.
11. Cristina Mano and Elena Rasa. A discussion of modeling techniques for personal lines pricing". Trans 27th ICA, 2002.
12. Mario V. Wüthrich and Michael Merz. Generalized linear models. ASTIN Bulletin: The Journal of the IAA, 49:1–3, 2019.

13. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

14. Andrea Gabrielli, Ronald Richman, and Mario V. Wüthrich. Neural network embedding of the over-dispersed poisson reserving model. Scandinavian Actuarial Journal, 2020(1):1–29, 2020. doi: 10.1080/03461238.2019.1633394. URL https://doi.org/10.1080/03461238.2019.1633394.

15. Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. ACM Comput. Surv., 51(5), August 2018. ISSN 0360-0300. doi: 10.1145/3236009. URL https://doi.org/10.1145/3236009.

16. Christoph Molnar. Interpretable Machine Learning. Creative Commons License, 2020.
17. Paul Embrechts and Mario V. Wüthrich. Recent Challenges in Actuarial Science. ULR https://docslib.org/doc/353687/recent-challenges-in-actuarial-science

18. Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. Introduction to linear regression analysis, volume 821 of Wiley series in probability and statistics. Wiley, Hoboken, 5th ed edition, 2012. ISBN 0470542810.

19. Nelder JA, Wedderburn RWM. 1972. Generalized linear models. J. R. Stat. Soc. Ser. A 135:370–84

20. McCullagh P, Nelder JA. 1983. Generalized Linear Models. London: Chapman & Hall

21. Lee SCK, Lin XS. 2010. Modeling and evaluating insurance losses via mixtures of Erlang distributions. N. Am. Actuar. J. 14:107–30

22. Miljkovic T, Grün B. 2016. Modeling loss data using mixtures of distributions. Insur. Math. Econ. 70:387–96

23. Yin C, Lin XS. 2016. Efficient estimation of Erlang mixtures using iSCAD penalty with insurance application. ASTIN Bull. 46:779–99

24. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http: //www.deeplearningbook.org.

25. Package «CASdatasets». ULR Package CASdatasets (uqam.ca)

26. Закон України про страхування. https://zakon.rada.gov.ua/laws/show/85/96-%D0%B2%D1%80#Text