Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення машинного навчання та обробки статистичних даних мовою Wolfram Language

Виконав: студент 6 курсу, групи СПм-61
спеціальності 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник (підпис) Бойко І.В. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) . (прізвище та ініціали)

Тернопіль 2023

АНОТАЦІЯ

Атестаційна робота магістра на тему «Розробка програмного забезпечення машинного навчання та обробки статистичних даних мовою Wolfram Language», Пешкович Сергій Ростиславович.

Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–61, Тернопіль, 2023. С. – 101, рис. – 52, табл. – 2, додат. – 2, бібліог. – 51.

Метою атестаційної роботи магістра є розробка та практична реалізація методів машинного навчання й засобів спрямованих на обробку статистичної інформації різної етимології на основі середовища Wolfram Mathematica та мови програмування Wolfram Language.

Технології розробки: Wolfram Mathematica, Wolfram Language, C#.

Технології для тестування: WolframUnit.

Розроблені програмні засоби та методології їх застосування мають безпосереднє спрямування у предметній області науки про дані, статистики, прикладної математики, людино-машинної взаємодії й призначені для практикуючих спеціалістів – програмістів та науковців працюючих у цих та суміжних галузях технологічної діяльності.

Ключові слова: статистична обробка даних, машинне навчання, лінійна регресія, логістична регресія, Wolfram Mathematica.

ABSTRACT

Master's certification work on the topic « Development of machine learning and statistical data processing software using the Wolfram Language », Peshkovich Serhii Rostislavovych.

Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Department of Software Engineering, SPm-61 Academic Group, Ternopil, 2023. P. – 101, fig. – 52, table. – 2, append. – 2, bibliog. – 51.

The purpose of the master's certification work is the development and practical implementation of machine learning methods and tools aimed at processing statistical information of various etymologies based on the Wolfram Mathematica environment and the Wolfram Language programming language.

Development technologies: Wolfram Mathematica, Wolfram Language, C#.

Technologies for testing: WolframUnit.

The developed software tools and methodologies for their application are directly related to the subject area of data science, statistics, applied mathematics, human-machine interaction and are intended for practicing specialists - programmers and scientists working in these and related areas of technological activity.

Keywords: statistical data processing, machine learning, linear regression, logistic regression, Wolfram Mathematica.

ЗМІСТ

ВСТУП

Нова вітка в розвитку науки про дані отримала своє життя, коли почалася пандемія COVID-19. Аналіз даних на той час займав чільне місце у засобах масової інформації: географічні карти показували особливо постраждалі райони (часто яскраво забарвлені в червоний колір), а графіки часових рядів ілюстрували збільшення та зменшення кількості підтверджених випадків або доступних лікарняних ліжок. Були зроблені прогнози з урахуванням епідеміологічних моделей, і такі терміни, як «базовий коефіцієнт відтворення», стали загальновідомими. Цей аналіз та прогнози по праву використовувалися як основа для важливих політичних рішень, оскільки вони надають переконливі кількісні докази.

Однак через свою статистичну природу такий аналіз дозволяє робити висновки та дії лише з високим ступенем невизначеності. Світ сформувався з настанням інформаційної доби у двадцять першому столітті. Значні досягнення в галузі штучного інтелекту відкривають шлях до його використання в повсякденному житті: пошукові системи переглядають Всесвітню павутину в пошуках відповідного контенту і намагаються вловити зміст даних, що вводяться користувачем. Алгоритми автоматично сортують фотографії за змістом зображень, а «розумна» побутова техніка стає все більш поширеною. Незалежно від того, чи говоримо ми про статистику, великі дані чи машинне навчання, не може бути жодних сумнівів у тому, що наука про дані продовжуватиме відігравати значну роль у найближчі десятиліття.

Мета дипломної роботи – сформулювати початкову методологію, яка дозволить розпочати подальшу ґрунтовну роботу у цій цікавій та перспективній галузі. Успішне вивчення науки про дані вимагає двох важливих факторів. По-перше, дослідники даних-початківці повинні отримати практичний досвід, осмислено «граючи» з даними, наприклад, програмуючи свої власні моделі і

виконуючи власні дослідження параметрів. Для цього в Інтернеті доступні численні навчальні посібники, приклади коду та набори даних. Другий важливий стовп - це всебічне знання методів та глибоке розуміння принципів та ідей, що лежать в основі методології науки про дані. Дана дипломна робота допомагає прийти до таких знань та розуміння.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Згідно з Міжнародною організацією зі стандартизації (ISO) [1], дані – це «переінтерпретоване подання інформації у формалізованій формі, придатне для передачі, інтерпретації чи обробки». Мерріам-Вебстер [2] дає іншу характеристику: дані - це «фактична інформація (наприклад, вимірювання або статистика), яка використовується як основа для міркувань, обговорень або розрахунків».

Збір, обробка, інтерпретація та передача даних з метою отримання надійних та корисних знань – зазвичай за допомогою інформаційних технологій – є основною функцією науки про дані.

В емпіричних науках, таких як природничі науки, збирання та аналіз даних вже давно стали важливою частиною отримання знань. Сучасна фізика, наприклад, навряд чи була б мислимою без взаємодії теорії та експерименту: відхилення від експериментальних даних оголюють межі теоретичних моделей. Наприклад, одним із успіхів ньютонівської небесної механіки став точний опис та передбачення руху планет та інших небесних тіл у Сонячній системі. Однак теорія не могла пояснити орбітальні обурення Меркурія [3], які з високим ступенем точності були виміряні в дев'ятнадцятому столітті Урбан Левер'є. Ці орбітальні обурення можна було пояснити лише згодом з допомогою теорії гравітації Ейнштейна, загальної теорії відносності.

У контексті бізнесу збір та аналіз даних також стають все більш важливими. Дані можна розглядати як сировину, з якого виробляються знання, як економічний товар. Аналіз даних допомагає керівництву приймати рішення та розуміти, який вплив вони мають на бізнес (бізнес-аналітика, бізнес-аналітика). Наприклад, патентні дані можуть використовуватися для демонстрації інноваційної та мережевої діяльності різних організацій у конкурентному середовищі, інформуючи стратегічні рішення в галузі інновацій чи управління технологіями [4, 5].

Демографія - ще одна дисципліна, де аналіз даних необхідний. Наприклад, статистичні дані використовуються для опису регіональних відмінностей у поведінці людей у галузі мобільності (див. [6, 7]) та для надання дієвої інформації для міського планування.

Результати аналізу даних можуть бути передані людям через мову, а також через візуальні та графічні уявлення. На наступній діаграмі показані дані тимчасового ряду температури повітря, виміряної на висоті двох метрів над землею в районі Тегеля в Берліні, Німеччина, за 2018 рік [8]:

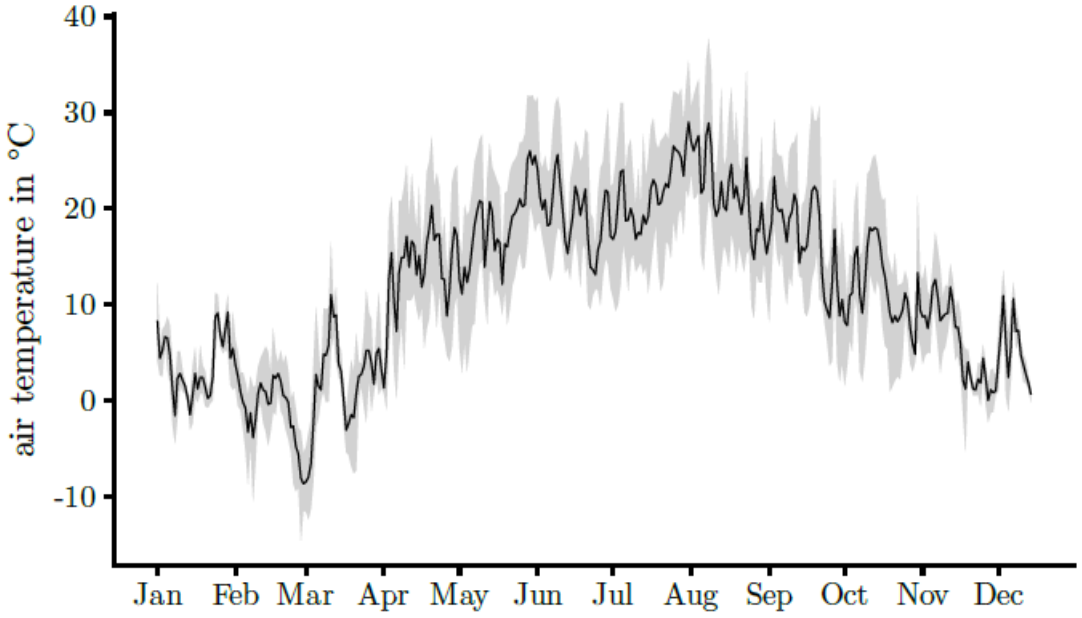


Рис. 1. Лінійна діаграма зміни температури

Сіра стрічка являє собою добову зміну мінімальної та максимальної температури. Легко побачити такі результати:

- Найхолодніші дні цього року були наприкінці лютого та на початку березня.
- Очевидно, що взимку холодніше, ніж улітку. Однак діаграма також показує, наскільки сильніше розрізняються денні та нічні температури влітку порівняно із зимою. Слід звернути увагу, що сіра стрічка влітку набагато ширша, ніж узимку.

Ту ж інформацію не можна було так легко та ефективно отримати з таблиці вимірювань. Підобласть аналізу даних, у якій основна увага приділяється графічному уявленню, називається візуалізацією даних.

Інший спосіб передачі даних - перетворення даних на акустичні нелінгвістичні сигнали - процес, званий ультразвуковою обробкою. Однак цей процес зустрічається набагато рідше, ніж графічне уявлення.

Отже, підсумовуючи: метою аналізу даних є отримання актуальної інформації та корисних знань шляхом систематичної організації, статистичної обробки та/або графічного (далі: акустичного, аудіовізуального) подання даних (у даному контексті також званих необробленими даними).

Наступні форми аналізу даних характеризуються відповідними цілями:

- Описова статистика використовується для організації та узагальнення даних.
- Дослідницький аналіз даних знаходить у них приховані закономірності для формулювання нових гіпотез.
- Інференційна статистика спрямована на опис спостережень шляхом їх зіставлення зі статистичними моделями та перевірки правдоподібності гіпотез, що ґрунтуються на даних.

Коли дослідницький аналіз даних виконується у великих масштабах, його називають інтелектуальним аналізом даних. Можливі описи інтелектуального аналізу даних - це "процес виявлення цікавих закономірностей з величезних обсягів даних" [9] або "практика пошуку у великих обсягах комп'ютеризованих даних для виявлення корисних закономірностей або тенденцій" [10].

Ще одним центральним завданням науки про дані сьогодні є розробка алгоритмів для інтелектуальних та автономних систем з використанням машинного навчання [11-20], щоб ці системи працювали за правилами, які не були явно задані, а в основному «вивчалися» на основі навчальних даних.

Медотологічно зазвичай описуються концепції та методи математики, статистики та обчислювальних наук [21-34], призначені для вирішення описаних вище задач. Така методологія в дипломній роботі буде поділена на три частини.

Перша частина присвячена аспектам організації даних: концептуальному та логічному структуруванню даних та способам забезпечення їх якості. У цій частині ми також представляємо набір інструментів описової статистики, метою яких є узагальнення та представлення основних характеристик зібраних даних.

Друга частина служить відноситься у математичну область стохастики, яка включає теорію ймовірностей та статистику виведення. Вони забезпечують концептуальну та методологічну основу для міркувань в умовах невизначеності.

Також ми дослідимо аспекти статистичної теорії навчання та представимо алгоритмічні методи машинного навчання, у яких суттєво використовуються концепції стохастики.

"Практика веде до досконалості" – показують результати роботи сучасних програмістів – спеціалістів з науки про дані. Велике значення має вибір корисно мови та середовища програмування, використовуючи мову програмування для статистичних обчислень на свій вибір. Сучасна сера науки про дані використовує наступні три мови програмування: R(18% ринку), Python(20% ринку) і Wolfram Mathematica (18% ринку). В методології використовуваній у дипломній роботі в основному застосовуватиметься мова Wolfram Mathematica та пов’язані з не парадигми програмування [35-51].

2. МАШИННЕ НАВЧАННЯ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ WOLFRAM

2.1. Загальні підходи до побудови методології машинного навчання мовою Wolfram. Алгоритм градієнтного спуску

В загальному випадку розвиток засобів машинного навчання слід розпочинати із розгляду підходів до градієнтного спуску як методу оптимізації для лінійної регресії; з виконанням відповідних обчислень, а також розгляду концепцій навчання розвинених моделей. Ми встановимо, як використовувати спеціалізовані функції мови Wolfram для машинного навчання, такі як Predict, Classify і ClusterClassify, у випадку лінійної регресії, для логістичної регресії та для кластерного пошуку. Крім того, для цих функцій будуть показані різні об’єкти та результати, які генерують ці функції, а також показники для вимірювання моделі. У кожному випадку слід виокремлювати, які частини моделі є фундаментальними для правильної побудови за допомогою мови Wolfram. Безпосередньо ми будемо використовувати приклади відомих наборів даних, таких як набір даних Fisher's Irises, набір даних про житло в Бостоні та набір даних Titanic.

Градієнтний спуск — це алгоритм оптимізації, який полягає у пошуку мінімуму функції у процесі ітеративного процесу. Для побудови процесу квадрат функції втрат помилок мінімізується за допомогою гіпотези лінійної моделі форми $f(xl) = \theta 0 + \theta_1 X_j$ в околі точки X_j. Функція втрат визначається таким виразом:

$$J(\theta) = \frac{1}{2 * N} * \sum_{j=1}^{N} \left(f\left(x_j\right) - y_j \right)^2$$

Ітераційний процес алгоритму складається з розрахунку коефіцієнтів до досягнення збіжності. Коефіцієнти задаються такими виразами:

$$\theta_0^{\,i+1} = \theta_0^{\,i} - \frac{\alpha}{N} * \sum_{j=1}^{N} \left(\theta_0^{\,i} + \theta_1^{\,i} * x_j - y_j \right)$$

$$\theta_1^{\,i+1} = \theta_1^{\,i} - \frac{\alpha}{N} * \sum_{j=1}^{N} \left(\theta_0^{\,i} + \theta_1^{\,i} * x_j - y_j \right) * x_j \,,$$

де підсумовування отримано з частинних похідних θ_0 і θ_1. Множник α відповідає швидкості навчання, яка є параметром, що мінімізує помилку при побудові процесу навчання [36-40].

Спочатку ми спочатку визначаємо наші дані за допомогою функції RandomReal і встановлюємо початкове число. Це необхідно для забезпечення відтворюваності даних у разі використання одного й того самого прикладу.

```
In[1]:=
SeedRandom[888]
x=RandomReal[{0,1},50];
y=-1-x+0.6*RandomReal[{0,1},50];
```

Далі проаналізуємо дані шляхом побудови двовимірної діаграми розсіювання.

```
In[2]:= ListPlot[Transpose[{x,y}],AxesLabel→{"X axis","Y axis"},
PlotStyle→Red]
Out[2]=
```

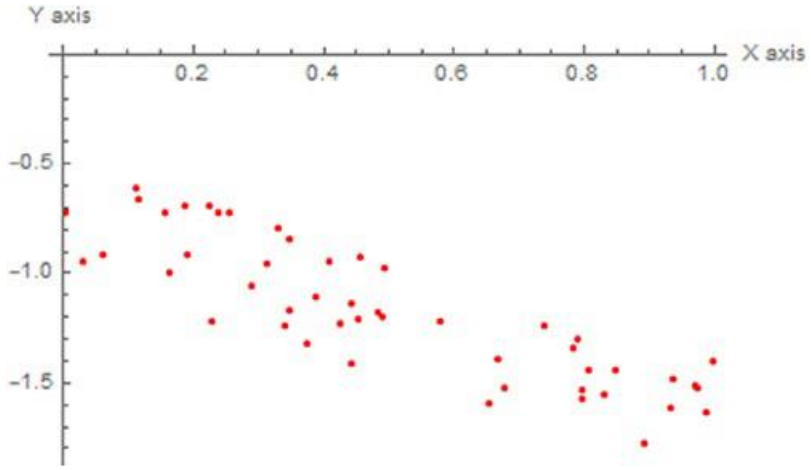


Рис. 2.1. 2D-діаграма розсіювання випадково згенерованих даних

2.2. Реалізація алгоритму градієнтного спуску

Тепер перейдемо до реалізації алгоритму на мові Wolfram. Алгоритм складатиметься з визначення констант, кількості ітерацій і швидкості навчання. Потім ми створимо два списки, що містять початкові значення нуль, в яких будуть зберігатися значення коефіцієнтів для кожної ітерації. Пізніше ми виконаємо обчислення коефіцієнтів через цикл з таблицею, який не завершиться, поки ми не досягнемо кількості ітерацій. У нашому випадку ми встановимо кількість ітерацій 250 зі швидкістю навчання 1.

```
In[3]:= itt=250;(*Number of iterations*)
α=1;(*Learning rate*)
θ0=Range@@{0,itt};(* Array for values of Theta_0*)
θ1=Range@@{0,itt};(* Array for values of Theta_1*)
Table[{
θ0[[i+1]]=θ0[[i]]- α/Length@x x *Sum[(θ0[[i]]+θ1[[i]]* x[[j]]-y[[j]]),
{j,1,Length@x}];
θ1[[i+1]]=θ1[[i]]- α/Length@x x *Sum[( θ0[[i]]+θ1[[i]]*x[[j]]-
y[[j]])*x[[j]],{j,1,Length@x}];},{i,1,itt}];
```

Оскільки з розрахунком коефіцієнтів ми визначилися, побудуємо рівняння лінійного коригування, побудувавши функцію та використовуючи значення коефіцієнтів останньої ітерації, що знаходяться в останній позиції списків θ_0 у θ_1.

```
In[4]:= F[X_]:= θ0[[Length@ θ0]]+ θ1[[Length@ θ1]]*X
```

Щоб дізнатися про форму найкращої відповідності, ми додаємо змінну X як аргумент. Це надасть нам форму: $F(X) = \theta_0+\theta_1*X$

```
In[5]:= F[X]
Out[5]= -0.707789-0.923729 X
```

Давайте подивимося, як лінія відповідає даним Рис. 2.2:

```
In[6]:= Show[{Plot[F[X],{X,0,1},PlotStyle→Blue,AxesLabel→{"X axis",
"Y axis"}],ListPlot[Transpose[{x,y}],PlotStyle→Red]}]
Out[6]=
```

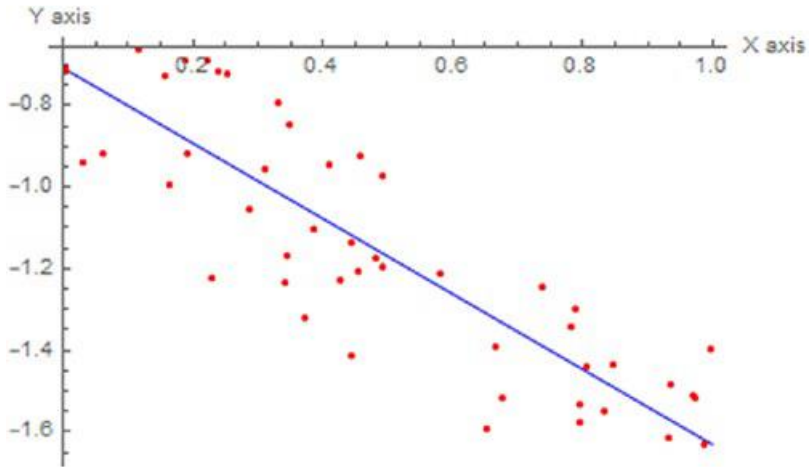


Рис. 2.2. Коригований лінійний список даних

Оскільки ми побудували лінійну модель, ми можемо здійснити графічне порівняння зміни швидкості навчання з кількістю ітерацій та значенням втрат, заданим функцією *J*.

Але спочатку ми повинні оголосити функцію втрат *J*. Для підсумовування ми можемо використовувати спеціальні символи сигми, або написати:

$$\sum_{i=1}^{imax} expr \text{ or Sum [expr, } \{i,i_{max}\}]$$

```
In[7]:=J[Theta0_,Theta1_]:=1/(2*Length[x])*Sum[(Theta0 + (Theta1*x[[i]]) -
y[[i]])^2,{i,1,Length@x}]
```

Нижче наведено графік втрат залежно від кожної взаємодії для значень швидкості навчання α1 = 1, α2 = 0,1, α3 = 0,01, α4 = 0,001 та α5 = 0,001 при повторенні процесу. Побачивши раніше збудований процес, ми можемо повторити процес для різних альф. Нижче наведено графік втрат залежно від кожної взаємодії для значень швидкості навчання α1 = 1, α2 = 0,1, α3 = 0,01, α4 = 0,001 та α5 = 0,001 при повторенні процесу.

```
In[8]:=
α1=Transpose[{Range[0,itt], J[θ0,θ1]}];
α2=Transpose[{Range[0,itt], J[θ0,θ1]}];
α3=Transpose[{Range[0,itt], J[θ0,θ1]}];
α4=Transpose[{Range[0,itt], J[θ0,θ1]}];
α5=Transpose[{Range[0,itt], J[θ0,θ1]}];
```

Побудуємо графік за допомогою ListLinePlot та візуалізуємо криву навчання для різних альф. Змінюючи значення альфа, будемо перевіряти, як зміниться і як налаштована лінія.

```
In[9]:= ListLinePlot[{α1,α2,α3,α4,α5},FrameLabel→{"Number of Iterations",
"Loss Function"},Frame→True,PlotLabel→"Learning Curve",PlotLegends→
SwatchLegend[{Style["α=1",#],Style["α=0.1",#],Style["α=0.01",#],Style[
"α=0.001",#],Style["α=0.0001",#]},LegendLabel→Style["Learning rate",White],
LegendFunction→(Framed[#,RoundingRadius→5,Background→Gray]&)]]&[White]
Out[9]=
```

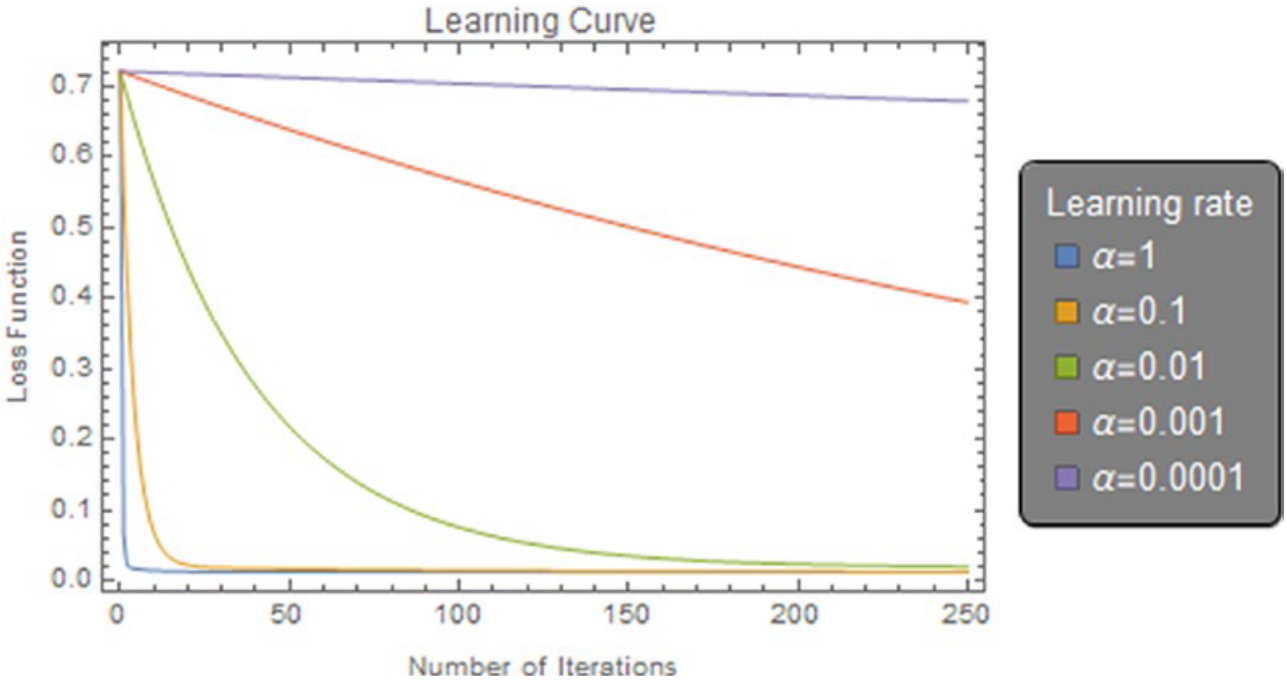


Рис. 2.3. Крива навчання для алгоритму градієнтного спуску

На попередньому графіку ми можемо візуалізувати розмір ітерацій відносно того, як вони змінюється залежно від значення альфа. Завдяки високій швидкості навчання ми можемо охопити більше значень на кожному етапі, але ризикуємо перевищити найнижчу точку. Щоб дізнатися, чи алгоритм працює, ми повинні бачити, що функція втрат зменшується на кожній новій ітерації. Інший випадок буде індикатором того, що алгоритм працює неправильно; це може бути пов'язано з різними факторами, такими як помилка коду або неправильне значення швидкості навчання. Як бачимо на графіку, адекватні значення альфа відповідають невеликим значенням у діапазоні від 1 до 10^{-4}. Немає необхідності

використовувати ті самі значення; Ви можете використовувати значення, що знаходяться у цьому діапазоні. Залежно від форми даних алгоритм може сходитися чи не сходитися з різними значеннями альфа, однаковими кроків ітерації. Якщо ми виберемо дуже малі значення альфа, алгоритму може знадобитися багато часу для збіжності, як бачимо для значень альфа 10^{-3} чи 10^{-4}.

2.3.Реалізаця алгоритму лінійної регресії. Набір даних Boston. Модель програмної системи регресії

Незважаючи на можливість створення алгоритмів для виконання лінійної регресії, мова Wolfram Language має спеціалізовану функцію для машинного навчання. У разі задач лінійної регресії є функція Predict. Функція Predict також може працювати з різними алгоритмами, а не лише алгоритмами задач регресії.

Функція Predict допомагає нам прогнозувати значення, створюючи функцію прогнозування з використанням навчальних даних. Це також дозволяє нам вибирати різні алгоритми навчання, які мають на меті прогнозувати числове, візуальне, категоріальне значення або їх комбінацію. Методи на вибір: дерево рішень, дерево з градієнтним посиленням, лінійна регресія, нейронна мережа, найближчі сусіди, випадковий ліс та гауссовий процес. До кожного методу є варіанти; варіанти різняться залежно від алгоритму, обраного на навчання функції прогнозування. Погляньмо на метод лінійної регресії. Вхідні дані Predict можуть бути у формі списку правил, асоціацій або набору даних.

Розглянемо завантаження даних Boston Homes з репозиторію даних Wolfram. Набір даних Бостон містить інформацію про житло в районі Бостон Массачусетс (Рис. 2.4). Безпосередньо інформацію також можна взяти із статті Девіда Харрісона та Деніела Рубінфельда «Гедоністичні ціни на житло та попит на чисте повітря», опубліковану в журналі. Журнал економіки та управління навколишнім середовищем.

```
In[1]:= Bstn=ResourceData[ResourceObject["Sample Data: Boston Homes"]]
Out[1]=
```

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0.00632	18	2.31	tract does not bound Charles river	0.538 ppm	6.575	65.2	4.09	1	296
0.02731	0	7.07	tract does not bound Charles river	0.469 ppm	6.421	78.9	4.9671	2	242
0.02729	0	7.07	tract does not bound Charles river	0.469 ppm	7.185	61.1	4.9671	2	242
0.03237	0	2.18	tract does not bound Charles river	0.458 ppm	6.998	45.8	6.0622	3	222
0.06905	0	2.18	tract does not bound Charles river	0.458 ppm	7.147	54.2	6.0622	3	222
0.02985	0	2.18	tract does not bound Charles river	0.458 ppm	6.43	58.7	6.0622	3	222
0.08829	12.5	7.87	tract does not bound Charles river	0.524 ppm	6.012	66.6	5.5605	5	311
0.14455	12.5	7.87	tract does not bound Charles river	0.524 ppm	6.172	96.1	5.9505	5	311
0.21124	12.5	7.87	tract does not bound Charles river	0.524 ppm	5.631	100	6.0821	5	311
0.17004	12.5	7.87	tract does not bound Charles river	0.524 ppm	6.004	85.9	6.5921	5	311
0.22489	12.5	7.87	tract does not bound Charles river	0.524 ppm	6.377	94.3	6.3467	5	311
0.11747	12.5	7.87	tract does not bound Charles river	0.524 ppm	6.009	82.9	6.2267	5	311
0.09378	12.5	7.87	tract does not bound Charles river	0.524 ppm	5.889	39	5.4509	5	311
0.62976	0	8.14	tract does not bound Charles river	0.538 ppm	5.949	61.8	4.7075	4	307
0.63796	0	8.14	tract does not bound Charles river	0.538 ppm	6.096	84.5	4.4619	4	307
0.62739	0	8.14	tract does not bound Charles river	0.538 ppm	5.834	56.5	4.4986	4	307
1.05393	0	8.14	tract does not bound Charles river	0.538 ppm	5.935	29.3	4.4986	4	307
0.7842	0	8.14	tract does not bound Charles river	0.538 ppm	5.99	81.7	4.2579	4	307
0.80271	0	8.14	tract does not bound Charles river	0.538 ppm	5.456	36.6	3.7965	4	307
0.7258	0	8.14	tract does not bound Charles river	0.538 ppm	5.727	69.5	3.7965	4	307

rows 1–20 of 506 columns 1–10 of 14

Data not in notebook; Store now »

Рис. 2.4. Набір даних про ціни на житло в Бостоні

Використаємо смуги прокручування, щоб отримати повне представлення набору даних. Розглянемо далі описи стовпців і покажемо їх у TableForm.

```
In[2]:= ResourceData[ResourceObject["Sample Data: Boston Homes"],
"ColumnDescriptions"]//TableForm
Out[2]//TableForm=
Per capita crime rate by town
Proportion of residential land zoned for lots over 25000 square feet
```

```
Proportion of non-retail business acres per town
Charles River dummy variable (1 if tract bounds river, 0 otherwise)
Nitrogen oxide concentration (parts per 10 million)
Average number of rooms per dwelling
Proportion of owner-occupied units built prior to 1940
Weighted mean of distances to five Boston employment centers
Index of accessibility to radial highways
Full-value property-tax rater per $10000
Pupil-teacher ratio by town
1000(Bk-0.63)^2 where Bk is the proportion of Black or African-American
residents by town
Lower status of the population (percent)
Median value of owner-occupied homes in $1000s
```

Ми створимо програмну модель, здатну прогнозувати ціни на житло у районі Бостона за кількістю кімнат у будинку. Для цього стовпці, що нас цікавлять, відповідають RM (середня кількість кімнат у житлі) і MEDV (середнє значення будинків, займаних власниками), оскільки ми хочемо з'ясувати, чи існує лінійна залежність між кількістю кімнат і ціна будинку. В результатів простого аналізу можна встановити, що будинки з найбільшою кількістю кімнат, можуть вмістити більше людей, і що це призводить до зростання цін. Почнемо із розгляду діаграми розсіювання MEDV та RM.

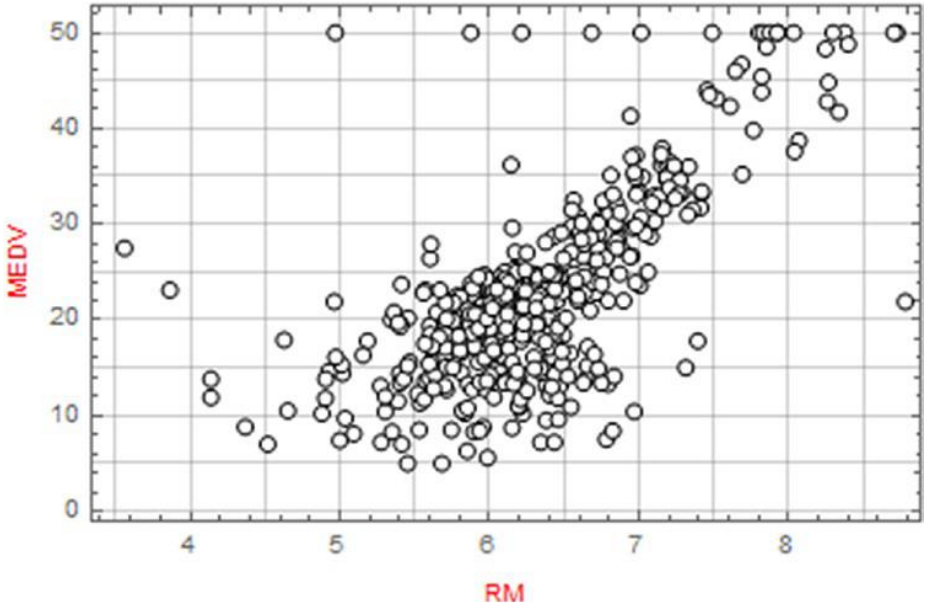


Рис. 2.5. 2D діаграма розсіювання

Як бачимо на Рис. 2.5, у міру збільшення середньої кількості кімнат зростає ціна будинку. Це говорить про те, що між цими двома змінними, можливо, існує прямо пропорційний зв'язок. Зважаючи на те, що видно на графіку, подивимося на значення кореляції між цими змінними. Ми покажемо це за допомогою кореляційної матриці, спочатку обчисливши кореляцію значень, призначивши імена тактів та побудувавши її на графіку за допомогою MatrixPlot (Рис. 2.6).

```
In[4]:= CorreLat=SetPrecision[Correlation[Transpose[{Normal[Bstn[All,"RM"]],
Normal[Bstn[All,"MEDV"]]}]],2];
XTicks={{1,"RM"},{2,"MEDV"},{1,"RM"},{2,"MEDV"}};
YTicks={{1,"RM"},{2,"MEDV"},{1,"RM"},{2,"MEDV"}};
PostionsValues={Text[#1,{0.5,1.5}],Text[#1,{1.5,0.5}],Text[#2,{1.5,1.5}],
Text[#2,{0.5,0.5}]}&[CorreLat[[1,1]],CorreLat[[1,2]]];
MatrixPlot[CorreLat,ColorFunction→"DarkRainbow",FrameTicks→{ XTicks,
YTicks,XTicks,YTicks},Epilog→{White,PostionsValues},PlotLegends→BarLegend
[{"DarkRainbow",{0,1}},4],ImageSize→180]
Out[4]=
```

Аналізуючи матричний графік (Рис. 2.6), можна дійти до висновку, що є хороший лінійний зв'язок між RM і MEDV.

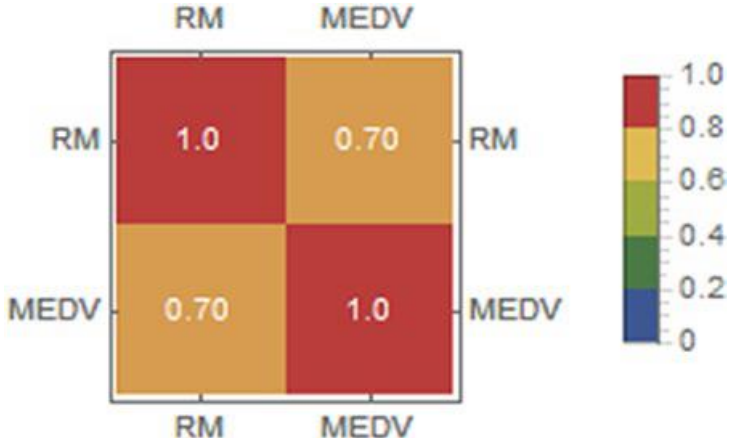


Рис. 2.6. Матричний графік у поєднанні з кореляційною матрицею

Тепер ми випадково перетасуємо набір даних і створимо список правил за допомогою Thread; це пов'язано з тим, що дані, які необхідно ввести в функцію прогнозування, мають бути такими: {x → y} — тобто вхідне та цільове значення.

```
In[5]:=
NewData=RandomSample[Thread[Normal[Bstn[All,"RM"]]→Normal[Bstn[All,
ssss"MEDV"]]]];
```

Для випадкової вибірки ми виберемо перші 354 елементи (70%), це буде навчальний набір, інші 152 (30%) будуть тестовим набором.

```
In[6]:={training,test}={NewData[[;;354]],NewData[[355;;]]};
```

Ми приступаємо до навчання моделі, передбачення середніх значень будинків, які займають власники (MEDV), як кінцевої мети. Як метод ми вибираємо лінійну регресію. При навчанні моделі специфікація варіанта звіту про навчання включає Panel (динамічний оновлення панелі), Print (періодична інформація, включаючи час, приклад навчання, найкращий метод, поточні втрати), ProgressIndicator (простий індикатор виконання), SimplePanel (динамічний оновлення). панель без графіків) та «None». Panel — опція за замовчуванням.

```
In[7]:=
PF=Predict[training,Method→"LinearRegression",TrainingProgressReporting→
"Panel"]
Out[7]=
```

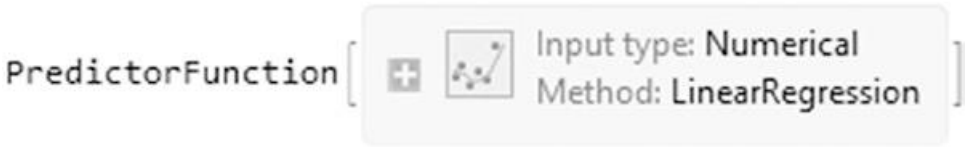


Рис. 2.7. Об'єкт PredictorFunction навченої моделі

При введенні коду, залежно від опції, доданої до TrainingProgressReporting, має з'явитися індикатор виконання та панельний звіт. Час відображення панелі залежить від часу навчання моделі. Щоб встановити конкретний час для навчання, додамо параметр TimeGoal, який визначає, як довго має тривати навчання моделі. Значення часу є секунди процесорного часу, тобто число без одиниць виміру. Для одиниць часу (секунди, хвилини та години) необхідно використовувати команду Quantity, наприклад TimeGoal → Quantity [“time magnitude”, #] & / @ {“Second”, “Minute”, “Hour”}.

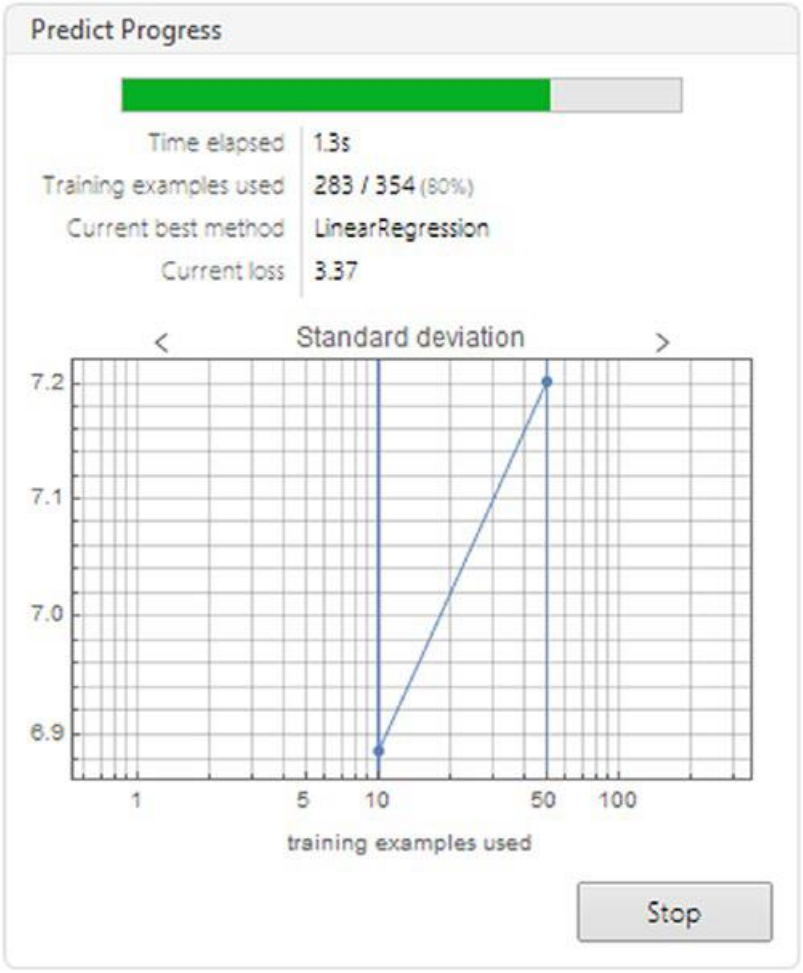


Рис. 2.7. Звіт про хід роботи PredictorFunction

Повернемося до моделі: як видно на Рис. 2.7, об'єкт, що повертається, є передикторною функцією (слід використовувати Head, щоб перевірити це). Надавши функції предиктора ім'я, для цього можна отримати додаткову інформацію про модель; використовується команда Information. Функція Information працює для будь-якого іншого виразу, а не лише для цілей машинного навчання.

```
In[8]:= Information[PF]
Out[8]=
```

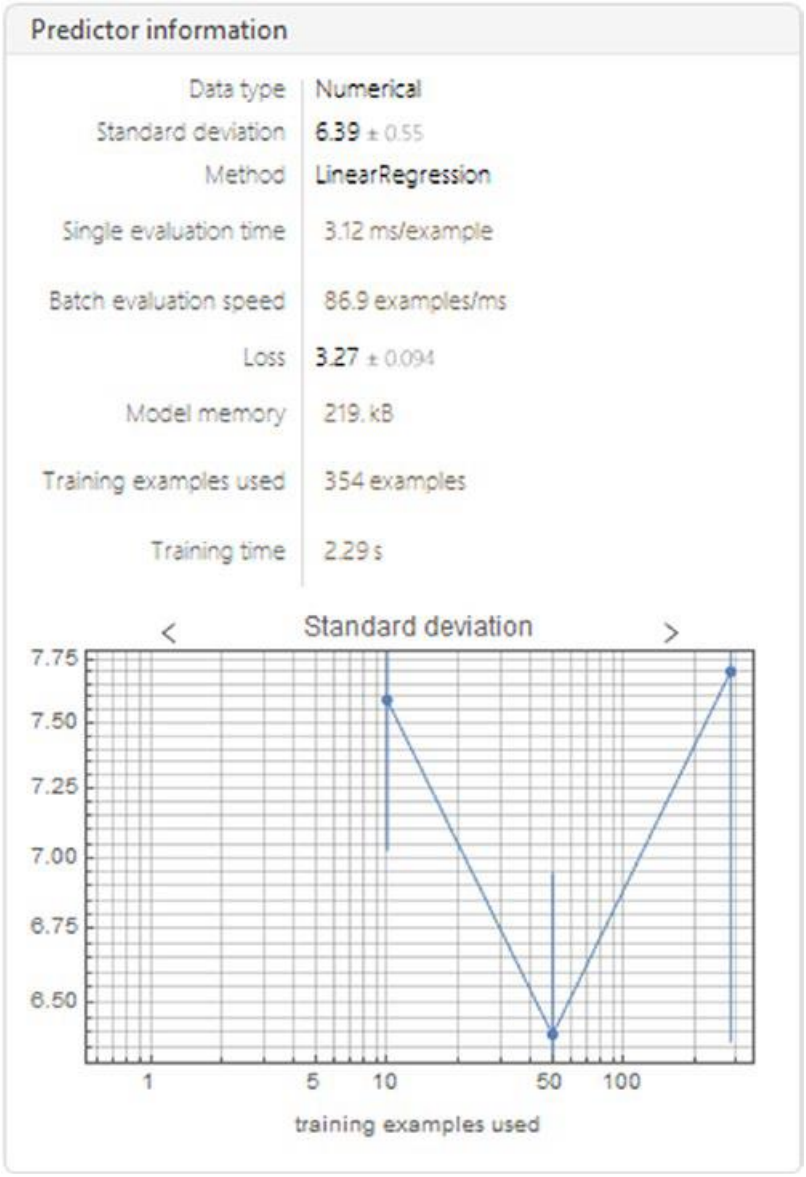


Рис. 2.8. Інформаційний звіт про стан навченої моделі

Інформаційна панель (Рис. 2.8) включає тип даних, середньоквадратичне відхилення (StandardDeviation), метод, швидкість пакетної оцінки, втрати, пам'ять моделі, кількість прикладів для навчання та час навчання. Графіки в нижній частині панелі відображають стандартне відхилення, криву навчання моделі та криву навчання інших алгоритмів. Якщо навести курсор на числові параметри, відобразяться довірчі інтервали та одиниці вимірювання. Якщо це зробити за назвою методу, будуть показані параметри методу лінійної регресії. Оскільки ми не вибрали конкретний алгоритм оптимізації в рамках методу лінійної регресії, Mathematica намагається знайти найкращий алгоритм (це можна побачити на

кривій навчання для всіх алгоритмів). Далі нам треба узагальнити, як отримати доступ до цих опцій.

У Табл. 2.1 представлені різні допустимі можливості, які можуть бути використані для моделювання тренувань, як добре, як їх визначення та можливі значення для тренування процесу PredictorFunction.

Таблиця 2.1 — Найпоширеніші параметри функції прогнозування

Опція	Визначення
Method	Алгоритм. Можливі значення: DecisionTree, GradientBoostedTrees, LinearRegression, NearestNeighbors. NeuralNetwork, Random_Forest і GaussianProcess.
PerformanceGoal	Оптимізація продуктивності. Можливі значення: DirectTraining, Memory, Quality, Speed, TrainingSpeed, Automatic. Підтримується поєднання значень (PerformanceGoal→ {val1, val2}).
RandomSeeding	Початкове значення для генератора псевдовипадкових чисел. Можливі значення: "Automatic". "custom seed", Inherited (випадкове початкове число, використане в попередніх обчисленнях).

Продовження таблиці 2.1

Опція	Визначення
TargetDevice	Вказання пристрою для виконання процесу навчання чи тестування. Можливі значення: CPU або GPU. Якщо встановлений графічний процесор, автоматичним цільовим пристроєм буде графічний процесор:
TimeGoal	Час, витрачений на навчальний процес
TrainingProgressIndicator	Звіт про виконання. Можливі значення: Panel, Print, ProgressIndicator, SimplePanel, None.

Після побудови моделі ми маємо спостерігати та аналізувати роботу функції прогнозування у тестовому наборі. Щоб це зробити, ми повинні зробити це за допомогою команди PredictorMeasurments. Як аргумент використовується передикторна функція (Рис. 2.9), за нею слідує тестовий набір, а потім властивість або властивості, які необхідно додати.

```
In[9]:= PRM=PredictorMeasurements[PF,test]
Out[9]=
```

Рис. 2.9. Об'єкт PredictorMeasurements тестованої моделі

Об'єкт, що повертається, називається PredictorMeasurementsObject (Рис. 2. 10). Ми також можемо додати властивості команди PredictorMeasurements. Ми можемо присвоїти об'єкту змінну, щоб спростити доступ до нього. Тепер проаналізуємо звіт роботи моделі із тестовим набором.

```
In[10]:= PRM["Report"]
Out[10]=
```

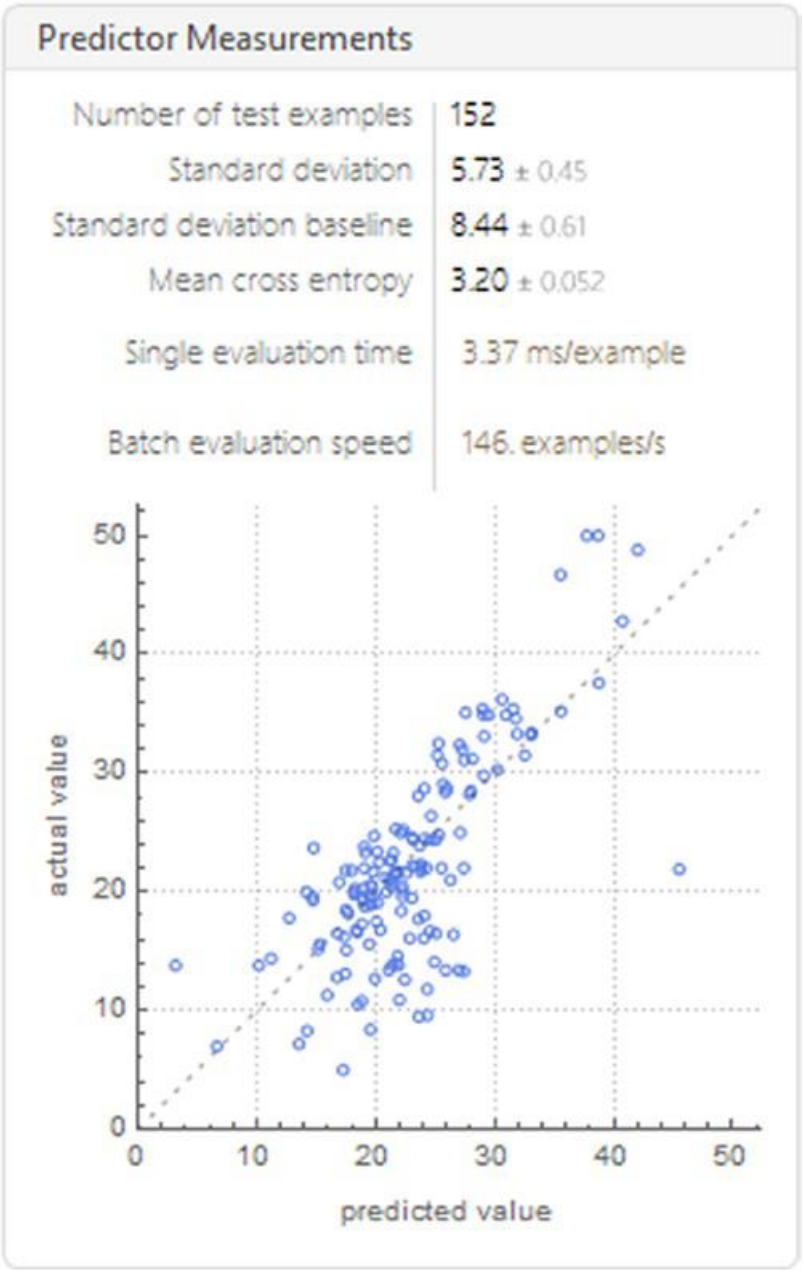


Рис. 2.10. Звіт роботи тестової моделі

У цьому звіті (рис. 2.10) показані різні параметри, такі як середньоквадратичне відхилення (стандартне відхилення), середня перехресна ентропія та інші. Він показує графік відповідності моделі разом із поточними та прогнозованими значеннями. Ми бачимо, що модель хороша для більшості випадків, за винятком того, що все ж таки є деякі викиди, що впливають на продуктивність.

Щоб краще зрозуміти точність моделі, погляньмо на середньоквадратичну помилку (RMSE) і RSquared (коефіцієнт детермінації), показані на Рис. 2.11. Щоб

відобразити пов'язані невизначеності, скористаємось опцією ComputeUncertainty зі значенням true.

```
In[11]:= Dataset[AssociationMap[PRM[#,ComputeUncertainty→True]&,{
"StandardDeviation","RSquared"}]]
Out[11]=
```

StandardDeviation	5.7 ±0.4
RSquared	0.54 ±0.10

Рис. 2.11. Стандартне відхилення та значення *r*-квадрату

Це дає нам трохи високе значення RMSE, а не хороше значення *r*-квадрату. Слід зауважити, що значення *r* у квадраті показує, наскільки хороша модель прогнозування. Ці два значення вказують на те, що хоча між кількістю номерів і цінами може існувати лінійна залежність, вона не обов'язково пояснюється лінійною регресією. Ці спостереження також узгоджуються, якщо згадати, що ми набули значення кореляції 0,7.

2.4. Підходи до оцінки ефективності моделей

Графіки, побудовані в моделі, є графіком моделі і цільовою змінною (ComparisonPlot). Щоб перевірити розподіл дисперсії, використовуємо функцію ResidualHistogram, а для перевірки графіка залишків - ResidualPlot.

```
In[12]:= PRM[#]&/@{"ResidualHistogram","ResidualPlot","ComparisonPlot"}
Out[12]=
```

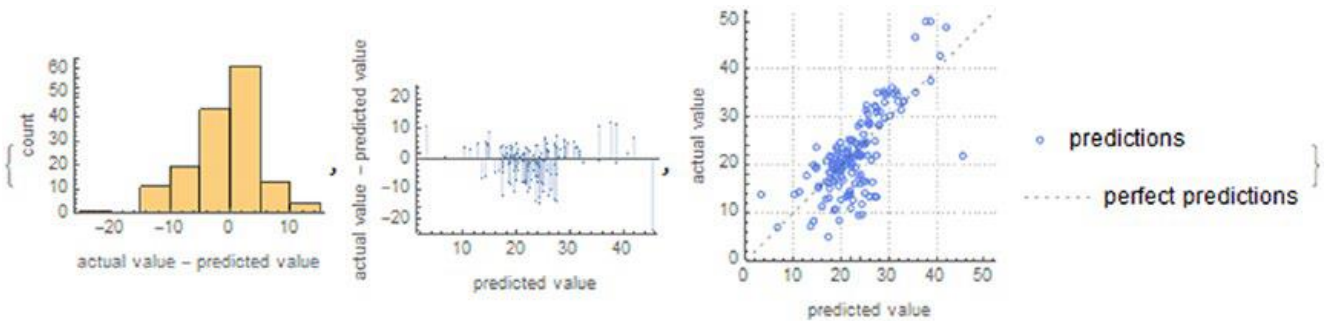


Рис. 2.12. ResidualHistogram, ResidualPlot, і ComparisonPlot

Щоб дізнатися про всі властивості об'єкта Predictor Measurements, ми підставляємо Properties як аргумент. Ці характеристики можуть відрізнятися залежно від методу.

```
In[13]:= PRM["Properties"]
Out[13]= {BatchEvaluationTime,BestPredictedExamples,ComparisonPlot,
EvaluationTime,Examples,FractionVarianceUnexplained,GeometricMeanProbability
Density,LeastCertainExamples,Likelihood,LogLikelihood,MeanCrossEntropy,
MeanDeviation,MeanSquare,MostCertainExamples,Perplexity,PredictorFunction,
ProbabilityDensities,ProbabilityDensityHistogram,Properties,RejectionRate,
Report,ResidualHistogram,ResidualPlot,Residuals,RSquared,StandardDeviation,
StandardDeviationBaseline,TotalSquare,WorstPredictedExamples}
```

Якщо нас не влаштовують вибрані методи або гіперпараметри, можна виконати перенавчання моделі шляхом налаштування нових значень гіперпараметрів. Ми отримуємо доступ до значень поточного методу за допомогою команди Information і додаємо властивості Method (показує нам метод, що використовується для навчання моделі), MethodDescription (опис методу, що використовується) і MethodOption (параметри методу).

```
In[14]:= Information[PF,"MethodOption"]
Out[14]= Method→{LinearRegression,L1Regularization→0,L2Regularization→
0.00001,OptimizationMethod→NormalEquation}
```

Як бачимо, існують такі терміни, як L1Regularization, L2Regularization і OptimizationMethod. Перші два терміни пов'язані з методами регуляризації, а L1 відноситься до імені регресії Лассо, а L2 - до імені регресії Ріджа. Регуляризація використовується для мінімізації складності моделі, а також зменшення варіацій; це також підвищує точність моделі, вирішуючи проблеми перенавчання.

Це досягається шляхом додавання штрафу до функції втрат; цей штраф додається до суми абсолютного значення коефіцієнта $\lambda_1 \sum_{i=0}^{N} |\theta_i|$ де для L2 він задається виразом $(\lambda_2/2)\sum_{i=0}^{N} \theta_i^2$, де функція, яку потрібно мінімізувати, є функцією втрат $(1/2)\sum_{i=0}^{N} \left(y_i - f(\theta, x_i)\right)^2$. Третій доданок — це вибір методу оптимізації, який хочемо вибрати; Існуючі методи - NormalEquation, StochasticGradientDescent і OrthantWiseQuasiNewton. Тим не менш, слід підкреслити, що при використанні вектора коефіцієнтів зі стандартами L1 та L2 це називається регресійною моделлю Elastic Net. Elastic Net можна використовувати у випадках, коли існує кореляція параметрів.

2.5. Методологія перенавчання гіперпараметрів моделі

Як обговорювалося пізніше, давайте перенавчимо модель, але зі значеннями L1 → 12, L2 → 100 і алгоритмом оптимізації OptimizationMethod → StochasticGradientDescent, TrainingProgressReporting → None, PerformanceGoal → «Quality», RandomSeeding → 1000.

```
In[15]:= PF2=Predict[training,Method→{"LinearRegression","L1Regularization"→
12,"L2Regularization"→100,"OptimizationMethod"→ Automatic},TrainingProgress
Reporting→None,PerformanceGoal→"Quality",RandomSeeding→10000,TargetDevice→
"CPU"];
```

Щоб переглянути властивості, що стосуються прикладу, введемо властивості після вхідних даних для функції Predictor, наприклад, PF2["example", "Properties"].

Тепер давайте порівняємо продуктивність нової моделі, порівнявши графіки та показники, які будо отримано раніше.

```
In[16]:= PRM2=PredictorMeasurements[PF2,test];
PRM[#]&/@{"ResidualHistogram","ResidualPlot","ComparisonPlot"}
Dataset[AssociationMap[PRM2[#,ComputeUncertainty→True]&,{
"StandardDeviation","RSquared"}]]
Out[16]=
```

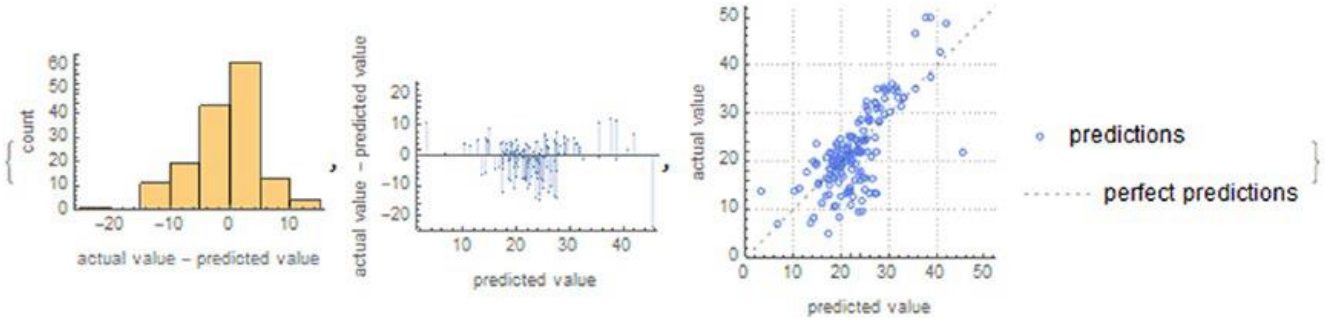


Рис. 2.13. Графіки перенавченої моделі

StandardDeviation	5.9 ± 0.4
RSquared	0.51 ± 0.10

Рис. 2.14. Нові значення для RMSE та *r* у квадраті

Аналізуючи графіки, бачимо, що модель лише певною мірою зменшується; це узгоджується з новим значенням r квадраті, яке зменшується до 0,51. Однак це, як і раніше, погана модель, коли справа доходить до прогнозування. Це можна пояснити вибором оптимізації, вибором параметрів L1 та L2.

3. СТАТИСТИЧНИЙ АНАЛІЗ ДАНИХ. МЕТОДИ ЛОГІСТИЧНОЇ РЕГРЕСІЇ

3.1 Випадкові числа та їх генерація

У цьому розділі ми розвинемо концепції та методи проведення аналізу за допомогою мови Wolfram Language, а також здійснимо лінійне коригування за допомогою рівнянь та реалізуємо спеціалізовані функції мови Wolfram Language для тієї ж мети. З використанням статистичних функцій. Мова Wolfram Language – корисний інструмент для статистики та теорії ймовірностей. У Mathematica є функції для виконання чисельних і наближених розрахунків для описової статистики та випадкових розподілів, випадкових чисел та методів випадкової вибірки.

Ми розглянемо основні команди для створення випадкових чисел — для цілих чисел, дійсних і комплексних. Функції для виконання випадкової вибірки із заміною та без заміни, , як зробити результати відтворюваними для випадкових чисел є предметом нашої роботи. Для створення випадкових чисел існує кілька функцій для створення випадкових цілих і дійсних чисел. Функція RandomInteger генеруватиме введені випадкові числа; якщо функцію не введені аргументи, інтервал генерації дорівнює 0 чи 1.

```
In[1]:= RandomInteger[]
Out[1]= 0
```

Щоб ввести діапазон, необхідно визначити його всередині функції; наприклад, між -1 та 1:

```
In[2]:= RandomInteger[{-1,1}]
Out[2]= 1
```

Щоб створити список випадкових чисел, нам потрібно визначити, скільки в списку ми хочемо.

```
In[3]:= RandomInteger[{-1,1},7]
Out[3]= {-1,0,1,-1,-1,-1,1}
```

Щоб числа повторювалися, як другий аргумент додамо форму списку або вкладений список. Наприклад, створимо вкладений список із семи елементів, у кожному підсписку буде чотири елементи.

```
In[4]:= RandomInteger[{-10,10},{7,4}]
Out[4]= {{-9,-2,6,10},{5,10,3,-7},{7,-2,-4,10},{-1,-2,8,-6},{-10,9,3,0},
{-4,-9,-2,5},{3,1,10,-5}}
```

Функція генерації випадкових чисел із десятковою точкою називається RandomReal. Він працює аналогічно RandomInteger, де інтервал записується між фігурними дужками.

```
In[5]:= RandomReal[]
Out[5]= 0.946141
```

Також є команда для комплексних випадкових і простих чисел.

```
In[6]:= RandomComplex[]
Out[6]= 0.411636 +0.79253 I
```

Для випадкових простих чисел ми повинні визначити мінімальний і максимальний інтервал, наприклад, якщо це просте число з перших 100.

```
In[7]:= RandomPrime[{1,100},6]
Out[7]= {43,11,61,83,61,79}
```

Цей тип функції генерує псевдовипадкові числа, тому можна встановити початкове число для генерації чисел. Це робиться за допомогою SeedRandom. За допомогою початкового числа ми можемо переконатися, що початкова послідовність випадкових чисел, що генеруються, однакова, щоб зробити випадкові вихідні дані відтворюваними. Щоб встановити початкове значення, використовуйте SeedRandom. Далі ми встановимо початкове число, за яким слідує послідовність випадкових чисел; як тільки насіння внесено, результати для цього насіння мають бути однаковими.

```
In[8]:= SeedRandom[6467789];RandomInteger[{-1,1},3]
Out[8]= {0,1,-1}
```

Для генерації результатів початкове число повинно бути в тому ж блоці коду. Існує можливість вибрати метод, як у наступному прикладі, де ми вибираємо метод MersenneTwister, який зазвичай використовується для створення

випадкових чисел. Використання іншого методу дає можливість генерувати послідовності різних випадкових чисел.

```
In[9]:= SeedRandom[Method→"MersenneTwister"];RandomInteger[{-1,1},{3,3}]
//MatrixForm
Out[9]//MatrixForm=
```

$$\begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & -1 \end{pmatrix}$$

Щоб повернутися до початкового значення, початкове число входить до функції без аргументів.

```
In[10]:= SeedRandom[]
```

Крім введення початкового числа, ми можемо створювати блоки випадкових чисел, в яких функції можна використовувати локально і не впливати на випадкову поведінку поза цими блоками. Це робиться за допомогою функції BlockRandom.

```
In[11]:= BlockRandom[RandomReal[1]]
Out[11]= 0.943218
```

Якщо ми запустимо алгоритм, який генерує випадкові числа всередині BlockRandom і оголосимо власне початкове число, це не повинно вплинути на інші процеси, в яких випадкові числа генеруються поза BlockRandom. Для ілюстрації розглянемо такий приклад.

```
In[12]:=
SeedRandom[121];
{RandomReal[],BlockRandom[RandomReal[]],RandomReal[],RandomReal[]}
Out[12]= {0.0908251,0.194288,0.194288,0.296762}
```

Як бачимо, останній процес генерував різні випадкові числа.

Щоб зробити вибірку із заміною, потрібно використовувати функцію RandomChoice. Щоб вибрати один елемент, ми пишемо лише список. Ми встановимо насіння, щоб отримати ті самі результати.

```
In[13]:= SeedRandom[12345];
RanData=RandomReal[{0,1},10]
Out[13]= {0.121246,0.329922,0.782753,0.430168,0.223586,0.463053,0.738017,
0.707618,0.790911,0.105714}
```

Ми згенерували список із 10 випадкових чисел у діапазоні від 0 до 1 і тепер приступаємо до випадкового вибору елемента з цих чисел.

```
In[14]:= RandomChoice[RanData]
Out[14]= 0.738017
```

Це дає нам єдиний результат зі списку із 10 елементів. Аналогічним чином ми можемо вибрати кількість вибірок з кількістю елементів у наступній формі: RandomChoice [«дані», «кількість вибірок», «кількість елементів»]. З 10 елементів ми виберемо три зразки з одним елементом.

```
In[15]:= RandomChoice[RanData,{3,1}]
Out[15]= {{0.329922},{0.738017},{0.223586}}
```

Хоча, якщо ми хочемо, щоб це було в тій самій вибірці, нам потрібно лише вказати кількість елементів для вибору.

```
In[16]:= RandomChoice[RanData,5]
Out[16]= {0.790911,0.463053,0.329922,0.430168,0.329922}
```

Щоб вибрати вибірку без заміни, використовуємо RandomSample. Ця функція не вибирає елемент списку зі списку даних більше одного разу. Для вибору ми вказуємо лише кількість елементів у вибірці як другий аргумент, оскільки перший відповідає списку даних.

```
In[17]:= RandomSample[RanData,9]
Out[17]= {0.105714,0.790911,0.463053,0.738017,0.707618,0.782753,0.430168,0.
223586,0.329922}
```

Придивившись до деталей, ми помічаємо, що значення, що повторюється, немає. Кожен елемент списку має рівну можливість бути обраним під час вибірки.

Щоб виконати системну вибірку, потрібно визначити розмір вибірки *M*. Щоб отримати розмір вибірки, ми можемо перерахувати елементи в списку або отримати довжину списку. Для початку ми створимо список із 200 простих чисел.

```
In[20]:= SeedRandom[09876]
RPrime=RandomPrime[{1,100},200];
Length[RPrime]
Out[20]= 200
```

Ми вже розрахували розмір вибірки, тому необхідно визначити розмір конкретної вибірки; у цьому випадку нам потрібна вибірка із 20 елементів. Після визначення вибірки розрахуємо інтервал позначеної вибірки j; j розраховується через співвідношення вихідного розміру вибірки, поділеного на загальну кількість елементів у зазначеній вибірці.

```
In[21]:= j=Length[RPrime]/20
Out[21]= 10
```

Це означає, що інтервал вибірки для нашої нової вибірки буде від 1 до 10. Звідси ми вибираємо випадкове число всередині інтервалу і звідти додаємо j разів, щоб вибрати наступний елемент; тобто для першого елемента це буде випадкове число діапазону h [1,10], для другого це буде $h + j$, а для третього $h + 3j$ і так далі, поки воно не досягне розміру вихідного зразка.

Ми вибрали випадкове число від 1 до 10.

```
In[22]:= RandomSample[Range[10],1]
Out[22]= {9}
```

Результат означає, що ми вибираємо із дев'ятого елемента. Ми розгортаємо список, щоб краще бачити дані.

```
In[23]:= RPrime
Out[23]= {17,11,67,97,11,73,71,61,71,31,59,29,79,7,71,89,79,11,2,29,97,61,
2,71,3,79,31,83,83,17,37,89,41,31,61,7,11,53,17,61,71,2,53,23,29,59,11,41,
13,71,3,53,13,61,19,2,17,17,59,3,11,41,83,59,41,47,13,59,17,5,5,59,79,37,
97,7,11,23,41,83,67,79,73,73,73,41,79,17,59,37,83,71,73,17,2,11,41,89,97,
7,2,23,13,67,79,83,5,61,47,73,61,97,53,53,2,89,19,19,61,89,83,43,73,3,83,
17,5,89,29,23,7,23,53,97,2,83,13,17,37,2,19,59,79,29,43,19,7,43,59,47,3,41,
23,53,37,59,29,83,37,59,19,59,31,89,2,67,47,47,97,2,47,97,41,11,43,37,7,59,
67,83,89,2,17,13,2,7,73,83,89,2,3,59,17,19,73,13,53,29,89,83}
```

Щоб отримати позиції елементів, що вибираються, це буде випадкове число для вибору, що дорівнює 9 плюс n разів j, поки у вас не буде 20 елементів.

```
In[24]:= Table[9+n*j,{n,0,19}]
Out[24]= {9,19,29,39,49,59,69,79,89,99,109,119,129,139,149,159,169,179,189,
199}
```

Ми маємо вибрати позиції, показані в попередньому висновку. Для вибору ми будемо використовувати позначення подвійної квадратної дужки.

```
In[25]:= Table[RPrime[[9+n*j]],{n,0,19}]
Out[25]= {71,2,83,17,13,59,17,41,59,97,47,61,29,37,59,37,97,67,89,89}
```

Далі краще розглянемо виділені елементи, виділивши їх червоним за допомогою MapAt і Style.

```
In[26]:= MapAt[Style[#,FontColor→ColorData["HTML"]["Red"]]&,RPrime,{#}&/@
{9,19,29,39,49,59,69,79,89,99,109,119,129,139,149,159,169,179,189,199}
Out[26]= {17,11,67,97,11,73,71,61,71,31,59,29,79,7,71,89,79,11,2,29,97,61,
2,71,3,79,31,83,83,17,37,89,41,31,61,7,11,53,17,61,71,2,53,23,29,59,11,41,
13,71,3,53,13,61,19,2,17,17,59,3,11,41,83,59,41,47,13,59,17,5,5,59,79,37,
97,7,11,23,41,83,67,79,73,73,73,41,79,17,59,37,83,71,73,17,2,11,41,89,97,7,
2,23,13,67,79,83,5,61,47,73,61,97,53,53,2,89,19,19,61,89,83,43,73,3,83,17,
5,89,29,23,7,23,53,97,2,83,13,17,37,2,19,59,79,29,43,19,7,43,59,47,3,41,23,
53,37,59,29,83,37,59,19,59,31,89,2,67,47,47,97,2,47,97,41,11,43,37,7,59,67,
83,89,2,17,13,2,7,73,83,89,2,3,59,17,19,73,13,53,29,89,83}
```

Як бачимо, системна вибірка не створює цілком випадкову вибірку. Випадковий вибір відбувається на першій стадії, коли ми вибираємо перший елемент створення нової вибірки. Після вибору першого елемента інші вибираються із послідовності невипадкових чисел. Ще один аспект, який слід враховувати, – це порядок вихідної вибірки; якщо між елементами існує періодичність, це може призвести до великої варіативності вибору елементів.

3.2. Загальні статистичні показники

Розуміння часто використовуваних статистичних формул має вирішальне значення розуміння того, як дані поводяться у різних умовах. Описова статистика реалізується після збору даних і є одним з перших кроків у процесі дослідного

аналізу даних, який дозволяє вам отримати уявлення про зібрані дані з точки зору виявлення закономірностей, аномалій, тенденцій, сезонності, варіацій тощо. Дослідницький аналіз даних є набір методів аналізу з метою виявлення характеристик, які не видно з першого погляду або не виявляються після збирання даних. Базова структура цього методу заснована на чисельному аналізі даних, графічному поданні та статистичній моделі. Багато причин для використання дослідницького аналізу даних включають перевірку даних, опис загальної і приватної ідеї базової структури, аналіз різних припущень, пов'язаних зі створенням моделі, і багато іншого.

Маючи вибірку даних, ми можемо розрахувати описові заходи. Заходи центрального тренду — це параметри, які дають інформацію про середні значення даних, підлягають вивченню. Середнє значення, також відоме як середнє арифметичне, є параметром, який розраховується із суми значень вибірки і поділу на суму кількості елементів. Функція Mean обчислює середнє значення.

```
In[27]:= List1=Table[Prime[i],{i,10}];
"Prime list :" <>ToString@ List1
"Mean: "<>ToString@Mean@N@List1
Out[27]= Prime list :{2, 3, 5, 7, 11, 13, 17, 19, 23, 29}
Out[27]= Mean: 12.9
```

Медіана – це значення, яке ділить вибірку на дві рівні частини, оскільки вона є середньою точкою даних, тому медіана – це значення симетрії щодо кількості даних. Функція Медіана дає це значення.

```
In[28]:= "Median: "<>ToString@Median@List1
Out[28]= Median: 12
```

Мода – найбільш поширене значення вибірки. Ми використовуємо команду Counts, яка дає нам кількість входжень кожного елемента до списку.

```
In[29]:= Counts[List1]
Out[29]= <|2→1,3→1,5→1,7→1,11→1,13→1,17→1,19→1,23→1,29→1|>
```

У цьому випадку входження дорівнює 1. Значення, що повторюються, немає; можна сказати, що у цій вибірці даних моди немає.

Вимірювання дисперсії дозволяють отримати інформацію про мінливість, представлену у зразку. Діапазон показує нам інтервал, де змінюються дані. Це виходить шляхом віднімання максимального значення та мінімального значення. Функції Max та Min повертають максимальне та мінімальне значення списку.

```
In[30]:= "Range: "<>ToString[Max[List1]-Min[List1]]
Out[30]= Range: 27
```

Дисперсія - це міра, отримана шляхом віднімання середнього значення кожного з елементів вибірки. Результат зводиться у квадрат із наступним додаванням. Сума ділиться на розмір вибірки. Його функція – дисперсія.

```
In[31]:= "Variance: "<>ToString[N[Variance[List1],3]]
Out[31]= Variance: 81.4
```

Стандартне відхилення — це вимір, отриманий із квадратного кореня дисперсії або за допомогою функції StandardDeviation.

```
In[32]:= {"Square root of Variance: " <>ToString[N[Sqrt[Variance[List1]],2]],
"StandardDeviation: " <>ToString[N[StandardDeviation[List1],2]]}
Out[32]= {Square root of Variance: 9.0,StandardDeviation: 9.0}
```

Стандартна оцінка - це оцінка, звана z, яка вимірює, на скільки стандартних відхилень відрізняється від середнього арифметичного для кожного елемента вибірки. Математичне рівняння має вигляд $z = (x - \mu)/\sigma$, де x - міра, μ - середнє значення, а σ - стандартне відхилення. Якщо z позитивне значення, це означає, що цей елемент більший за середній. Коли z негативно, це протилежний випадок. Ми визначимо z-оцінку для другого елемента у списку.

```
In[33]:= z=N[(List1[[2]]-Mean@List1)/StandardDeviation@List1,3];
"z score: " <> ToString@z
Out[33]= z score: -1.10
```

Це означає, що оцінка по другому елементу в 1,10 рази нижча за середню.

При розрахунку квартилю дані поділяються на чотири рівні частини. Нижній квартиль відповідає 25% квартилю даних, другий квартиль - 50%, третій квартиль (верхній квартиль) - 75% і четвертий квартиль (100%). Для розрахунку квартилів ми використовуємо функцію Quartiles, яка, своєю чергою, дає значення першого, другого та третього квартилю.

```
In[34]:= "Quartiles: " <> ToString@Quartiles[List1]
Out[34]= Quartiles: {5, 12, 19}
```

Якщо ми хочемо отримати одне значення, ми використовуємо функцію Quartile, за якою слідує відсоток, що обчислюється. Потім для розрахунку третього квартилю (75-го відсотка) ми використовуємо наступне:

```
In[35]:= Quantile[List1,0.75]
Out[35]= 19
```

Для розрахунку міжквартильного розмаху, який є різницею між верхнім і нижнім квартилем, використовується функція InterquartileRange.

```
In[36]:= InterquartileRange[List1]
Out[36]= 14
```

3.3. Статистичні діаграми та їх побудова у Wolfram Mathematica

Іноді, коли ми проводимо статистичне дослідження, можна знайти кількісні та якісні змінні; для цих змінних ми можемо створити гістограму. Гістограма (Рис. 3.1) є графічне подання, в якому на осі відображається кількість частот дискретної якісної змінної.

```
In[37]:= BarChart[{1,2,3,4},ChartLabels→{"feature 1","feature 2",
"feature 3","feature 4"}]
Out[37]=
```

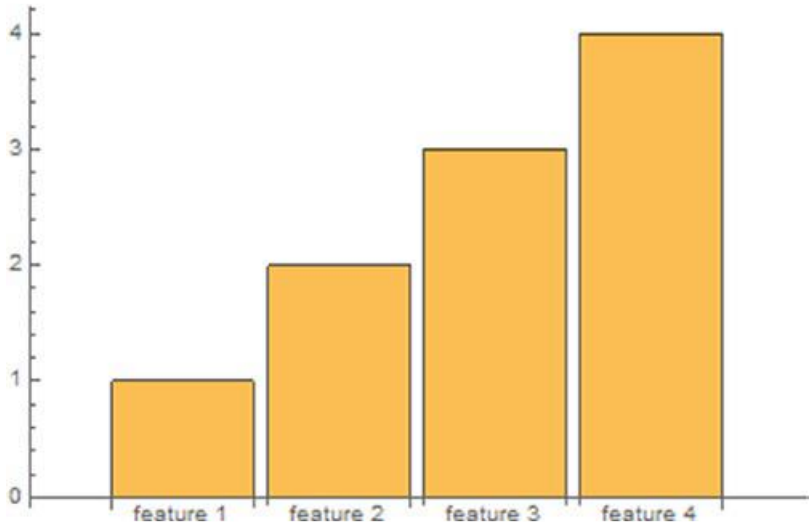


Рис. 3.1. Стовпчикова діаграма

Різні модальності якісної змінної розташовані на одній із осей. Інша вісь показує значення чи частоту кожної категорії у заданій шкалі. Смуга функції 2 має пов'язане значення 2. Орієнтація графіка може бути вертикальною, коли категорії розташовані на горизонтальній осі, а стовпці є вертикальними, або горизонтальною, коли категорії розташовані на вертикальній осі, а стовпці горизонтальні:

```
In[38]:= GraphicsRow[{BarChart[{1,2,3,4},ChartLabels→{"feature 1",
"feature 2","feature 3","feature 4"},BarOrigin→Bottom,ChartStyle→
LightBlue],BarChart[{1,2,3,4},ChartLabels→{"feature 1","feature 2",
"feature 3","feature 4"},BarOrigin→Left,ChartStyle→LightRed]}]
Out[38]=
```

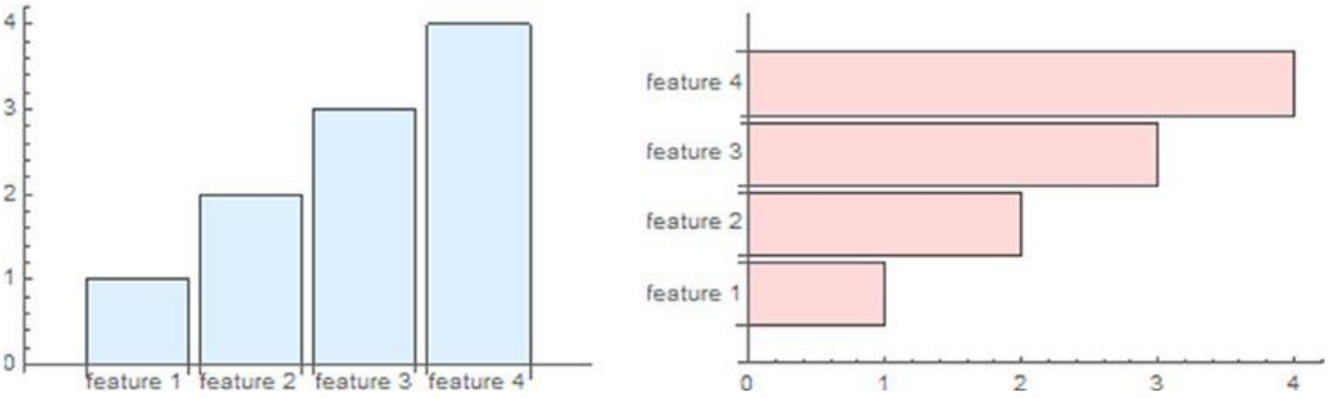


Рис. 3.2. Нижня та ліва вихідна гістограма

Гістограми можна використовувати для порівняння величин різних категорій та спостереження за тим, як значення змінюються в залежності від фіксованої змінної, наприклад кожного об'єкта. Крім того, ми можемо вибрати, як відображати стовпці: просто коли ми показуємо одну серію, як показано в попередньому прикладі; згрупований, що містить кілька рядів даних та представлений стовпцями різного типу; або складений, де смуга розділена на сегменти різного кольору, що представляють різні категорії. Розташування відсотків відображається у відсотковій шкалі, як показано на Рис. 3.3.

```
In[39]:= Labeled[GraphicsGrid[
{{
BarChart[{{4,3,2,1},{1,2,3},{3,5}},ChartLayout→"Grouped",ColorFunction→
"SolarColors"],
BarChart[{1,2,3,4},ChartStyle→LightRed,ChartLayout→"Stepped"]},
{BarChart[{{4,3,2,1},{1,2,3},{6,5}},ChartLayout→"Stacked"],
BarChart[{{4,3,2,1},{1,2,3},{6,5}},ChartLayout→"Percentile",ColorFunction→
"DarkRainbow"]
}},Frame→All,FrameStyle→Directive[Black,Dashed],Background→LightBlue,
ImageSize→500],"Bar Charts",Top]
Out[39]=
```

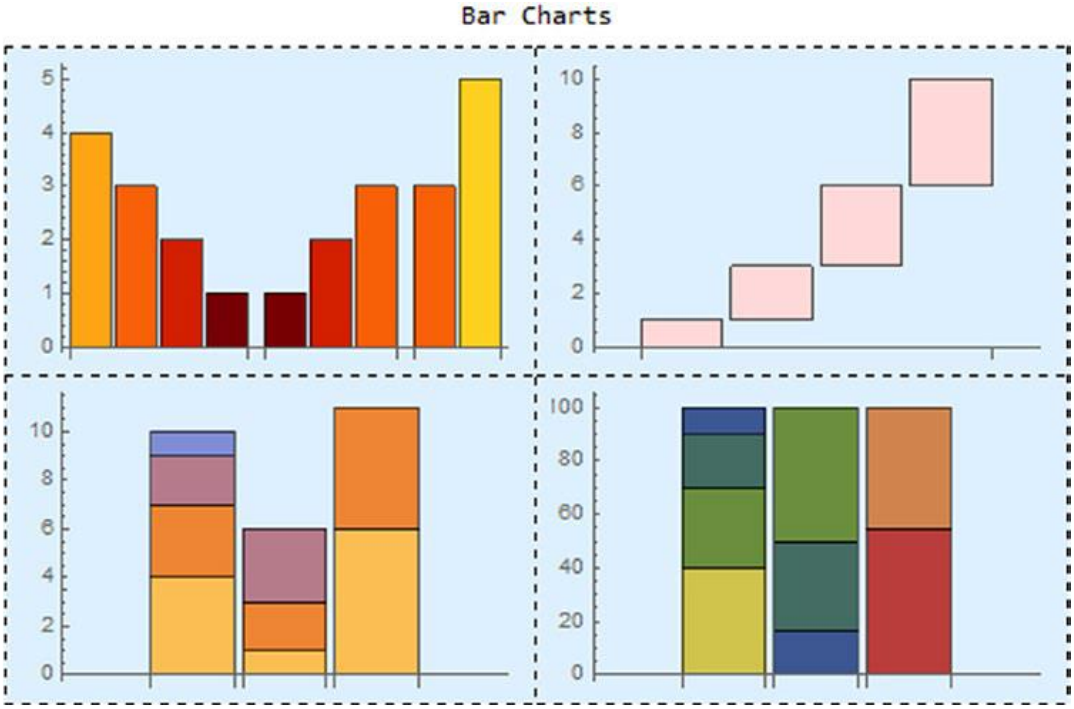


Рис. 3.3. Сітка гістограми

Існує також корисний аналог 3D-графіки BarChart3D.

```
In[40]:= SeedRandom[123]
Labeled[GraphicsGrid[
{{
BarChart3D[{{4,3,2,1},{1,2,3},{3,5}},ChartLayout→"Grouped",ColorFunction→
"SolarColors"],
BarChart3D[{1,2,3,4},ChartStyle→LightRed,ChartLayout→"Stepped"]},
{BarChart3D[RandomReal[1,{10,5}],ChartLayout→"Stacked"],
BarChart3D[{{4,3,2,1},{1,2,3},{6,5}},ChartLayout→"Percentile",
ColorFunction→"DarkRainbow"]
}},Frame→All,FrameStyle→Directive[Red,Thick],Background→LightBlue,
ImageSize→500],"3D Bar Charts",Top,Frame→True,Background→White]
Out[40]=
```

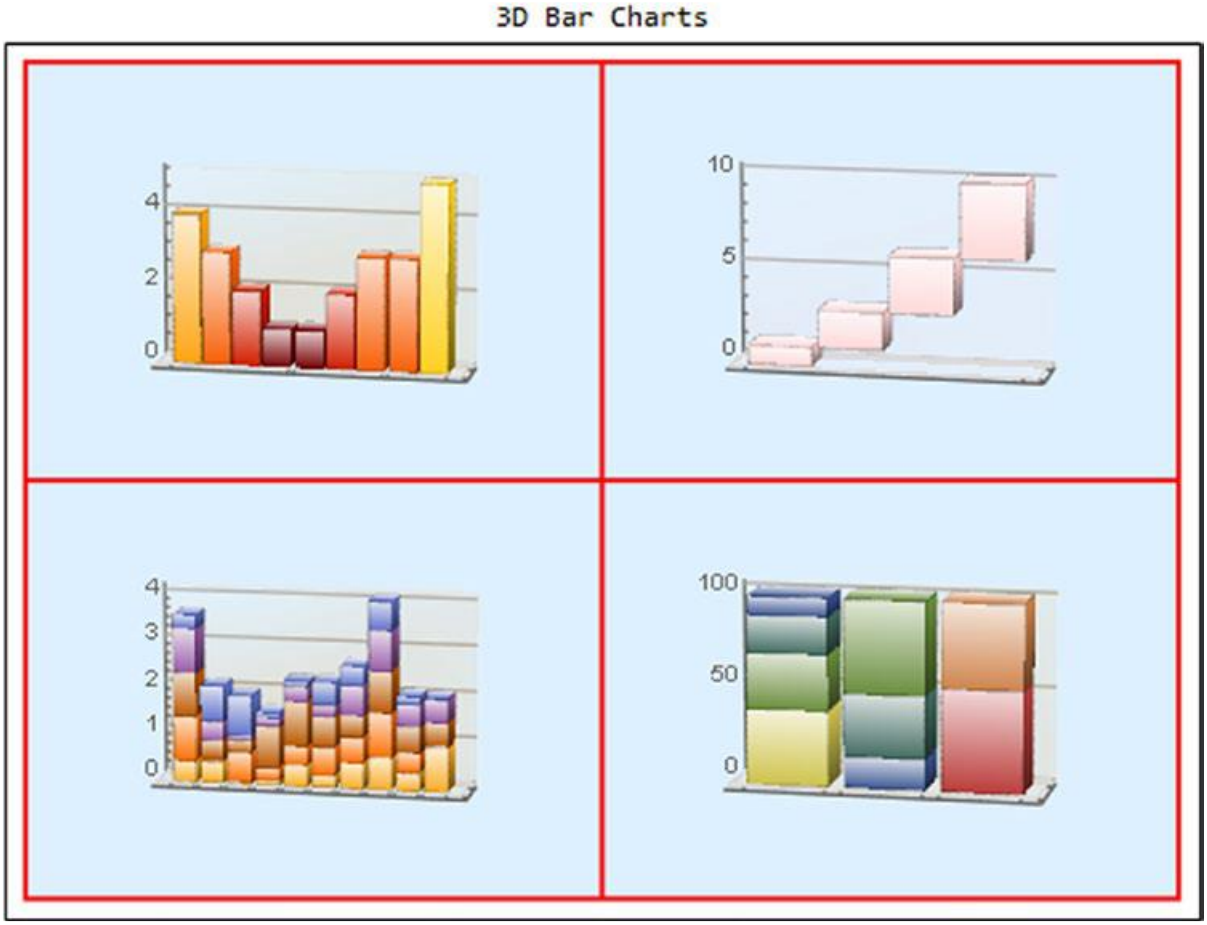


Рис. 3.4. Сітка тривимірних гістограм

Гістограми – це тип візуалізації, який зазвичай використовують у статистичних дослідженнях. За допомогою гістограми ми можемо побачити, як розподіляється вибірка. Гістограми використовуються для подання частот кількісної змінної. Класи змінної розташовані на горизонтальній осі, а частоти на

іншій осі. Далі ми побудуємо гістограму із сукупності 50 випадкових значень від 0 до 1 і встановимо кількість інтервалів, що дорівнює 10. Другий аргумент гістограми — визначена кількість інтервалів.

```
In[41]:= SeedRandom[4322]
hist1=Table[RandomReal[{2,3}],{i,0,20}];
Histogram[hist1,10]
Out[41]=
```

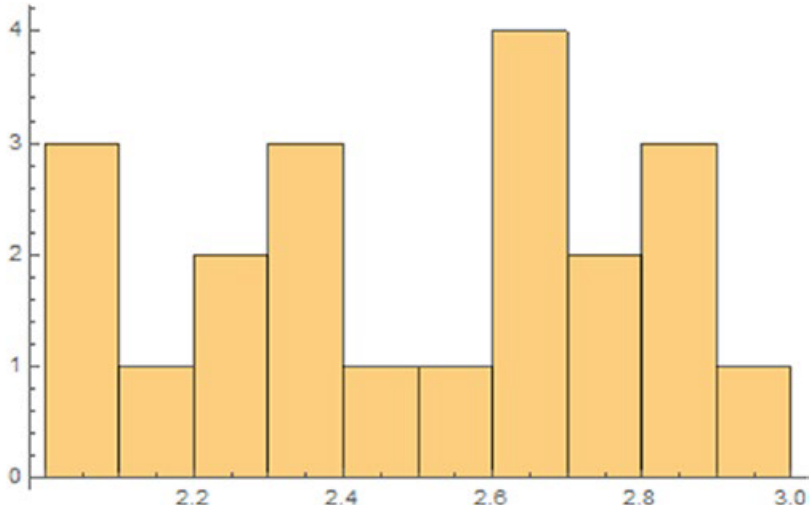


Рис. 3.5. Гістограма для випадкових дійсних чисел

Як і у випадку з лінійчастими діаграмами, існують способи редагування джерела гістограми, а також способу її відображення (накладення або накладення), як показано на Рис. 3.6.

```
In[42]:=
hist2=Table[Cos[i],{i,1,20}];
hist3=Table[Sin[i],{i,1,10}];
GraphicsColumn[{Histogram[{hist1,hist2},10,BarOrigin→Left,ChartStyle→
"Pastel",ChartLegends→{"rand num","Cos(x)"}],Histogram[{hist2,hist3},10,
ChartLayout→"Overlapped",ChartStyle→"Pastel",ChartLegends→{"Cos(x)",
"Sin(x)"}],Histogram[{hist2,hist3},10,ChartLayout→"Stacked",ChartStyle→
"Pastel",ChartLegends→{"Cos(x)","Sin(x)"}]}]
Out[42]=
```

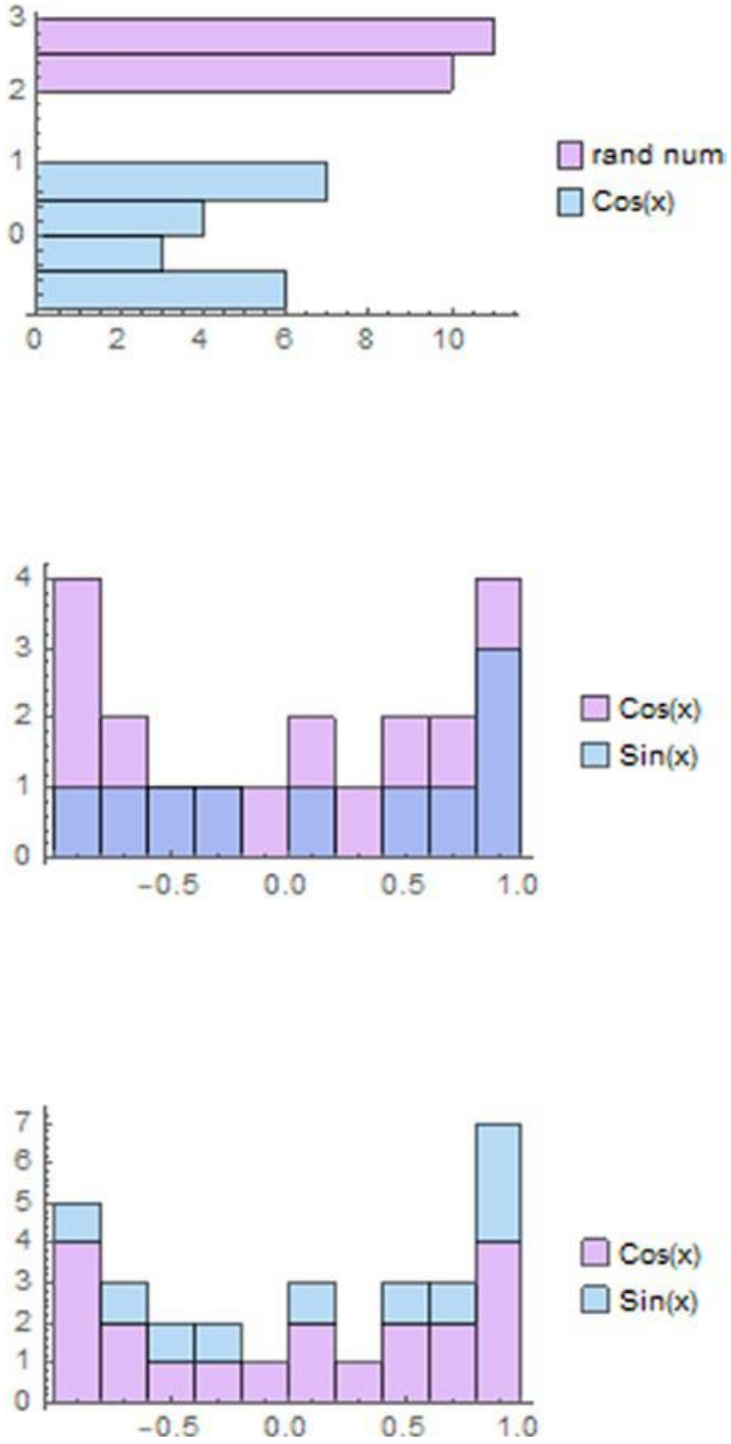


Рис. 3.6. Сітка форм гістограми

Маючи це на увазі, ми також можемо відображати двонаправлені гістограми, використовуючи PairedHistograms. Вони можуть мати горизонтальну або вертикальну орієнтацію та містити два ряди даних, стовпці яких йдуть у протилежних напрямках.

```
In[43]:= SeedRandom[123]
GraphicsRow[{PairedHistogram[{RandomReal[{0,1},20]},{RandomReal[{0,1},20]},
BarOrigin→Left],PairedHistogram[{RandomReal[{0,1},20]},{RandomReal[{0,1},
20]},10,BarOrigin→Top,ChartStyle→"Pastel"]}]
Out[43]=
```

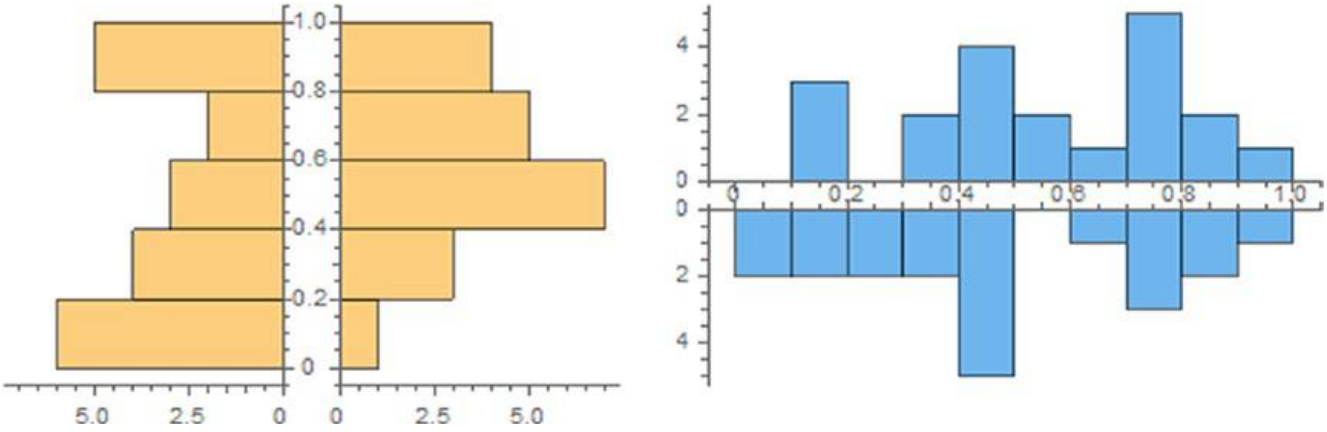


Рис. 3.7. Парні гістограми різного походження

Кругові діаграми є кола, поділені на дві або більше частин. Вони використовуються для подання кількісних змінних, які разом становлять суму; наприклад, розмір сектора зображується пропорційним вартості, яку він представляє, і виражається у відсотках, що дає лише відносну кількісну інформацію. Кругові діаграми створюються за допомогою команди PieChart.

```
In[183]:= GraphicsRow[{PieChart[{1,1,1},ChartLegends→{"part a","part b",
"part c"},ChartStyle→{LightRed, LightBlue, LightYellow}],
PieChart[{1,1},ChartLegends→{"part a","part b"},ChartStyle→"SunsetColors"]}]
Out[183]=
```

Рис. 3.8. Кругові діаграми

Секторні діаграми будуються за допомогою SectorChart (Рис. 39). Вони використовуються для порівняння різних даних, які зустрічаються в тому самому місці. Вони побудовані з урахуванням пропорційного розміру x значенню радіуса y. Розмірність, у якій виражаються величини, має бути однаковою всім сегментів.

```
In[45]:= SectorChart[{{2,1},{1,2}},ChartLegends→{"Sector a","Sector b"},
ChartStyle→{LightRed, LightYellow}]
Out[45]=
```

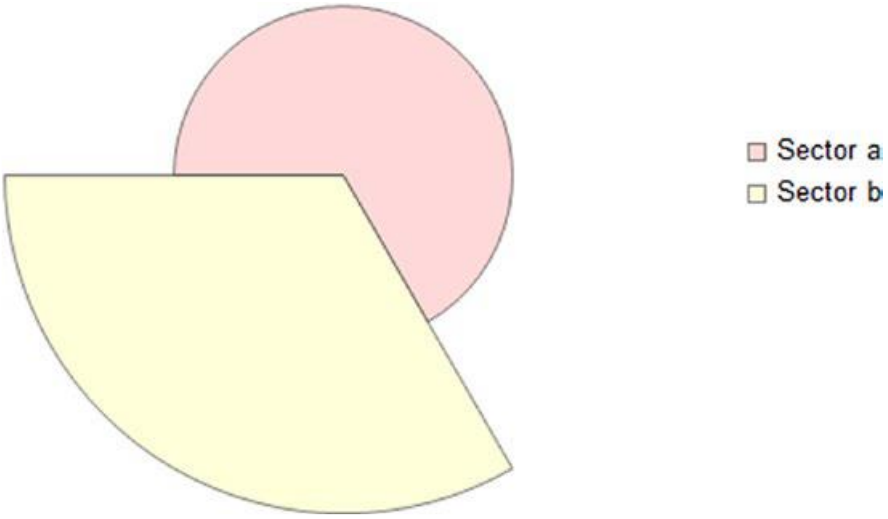


Рис. 3.9. Секторні діаграми

Для кожного видимого графіка існує відповідна команда для створення у трьох вимірах; такі графіки подано на Рис. 3.10.

```
In[46]:= GraphicsGrid[
{{SectorChart3D[{{2,1,1},{3,1,2},{1,2,2}},PlotLabel→"3D Sector chart",
ChartStyle→{Red, Blue,Yellow}],
PieChart3D[{1,1,1},ChartStyle→"GrayTones",PlotLabel→"3D Pie Chart"]},
{Histogram3D[Table[{i^3,i^-1},{i,20}],10,ChartElementFunction→
"GradientScaleCube",PlotLabel→"3D Histogram"],None}
},ImageSize→500,Frame→True,FrameStyle→Directive[Thick,Dotted]]
Out[46]=
```

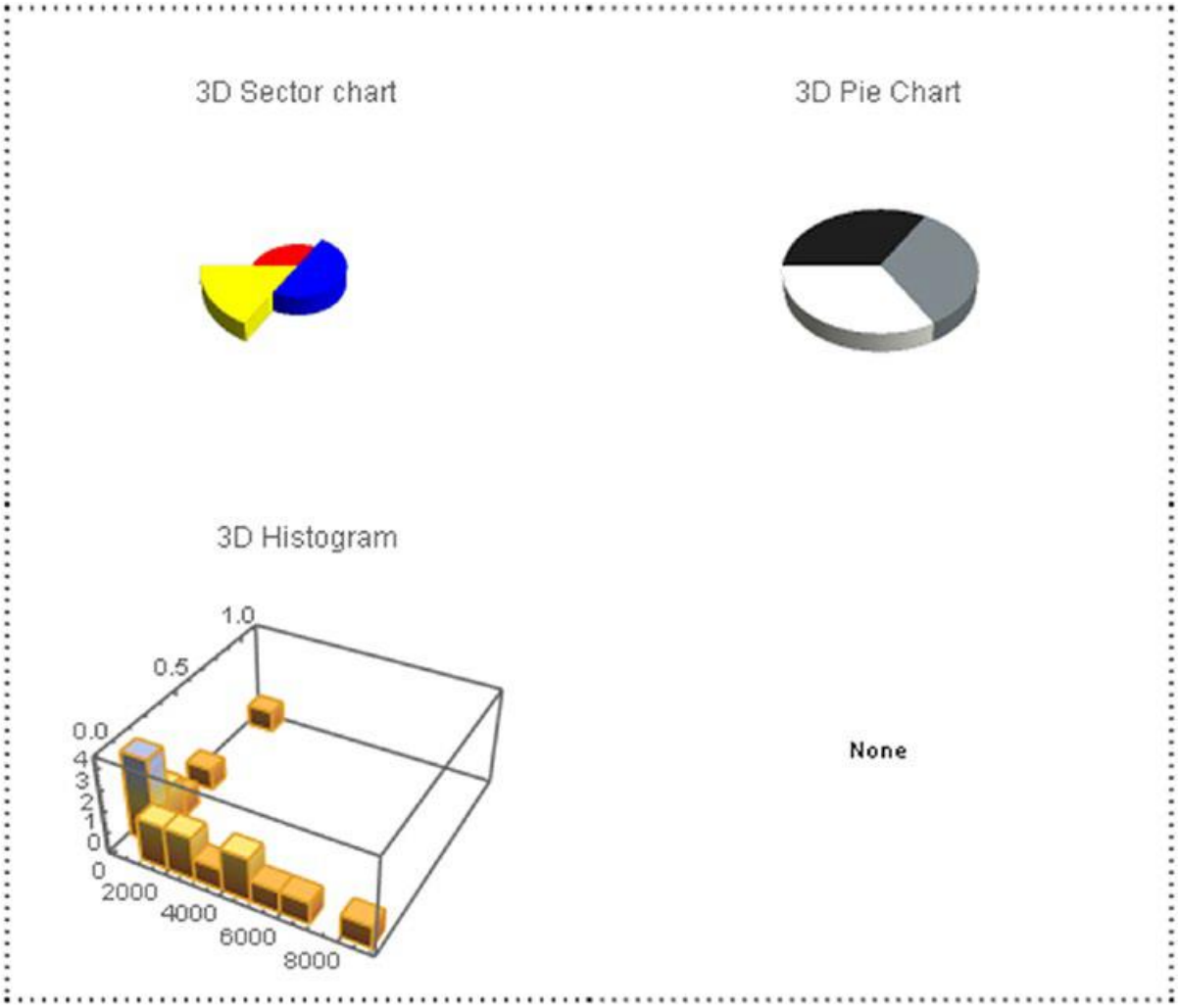


Рис. 3.10. 3D-сітчасті діаграми

Коробкова діаграма - це спосіб подання та спостереження за розподілом даних. Власне, він використовується виділення аспектів розподілу даних у одному чи кількох рядах. Для створення діаграми ми використовуємо команду BoxWhiskerChart.

```
In[47]:= SeedRandom[1234]
BoxWhiskerChart[{Table[RandomReal[],{i,0,50}],Table[RandomReal[],{i,0,50}],
Table[RandomReal[],{i,0,15}]},ChartLabels→{"Chart 1","Chart 2","Chart 3"}]
Out[47]=
```

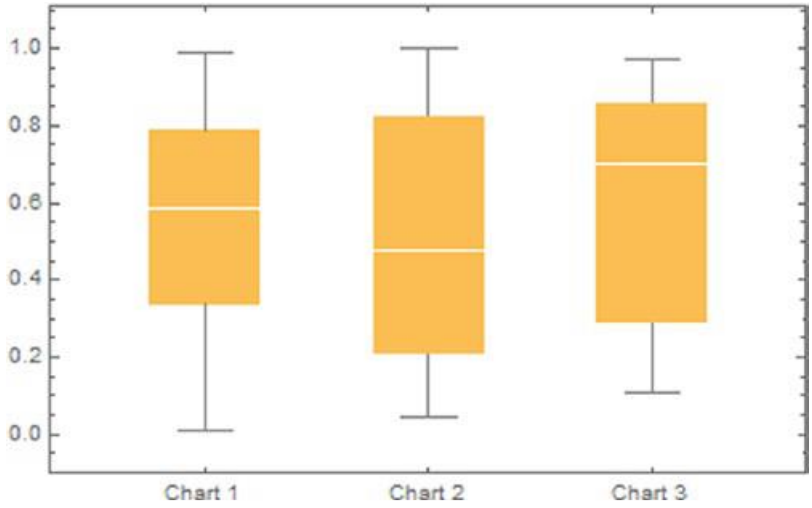


Рис. 3.11. Скринькова діаграма

Поле представлено прямокутником, що позначає міжквартильний діапазон розподілу. Перша лінія знизу нагору відзначає значення першого квартилю (25%), лінія, що перетинає рамку, - це медіана, а остання лінія, що обмежує рамку, - це третій квартиль (75%). Вуса - це лінії, що позначають максимальне та мінімальне значення. При наведенні курсору миші на графік буде показано інформацію про дані; сюди входять мінімум, максимум, медіана, 75-й процентиль та 1-й квартиль. Залежно від специфікації, що використовується нами, це може вплинути на те, які параметри і як будуть відображатися (Рис. 3.12).

```
In[48]:= SeedRandom[123]
GraphicsGrid[{{BoxWhiskerChart[{Table[RandomReal[],{i,0,50}],
Table[RandomReal[],{i,0,50}],Table[RandomReal[],{i,0,15}]},#,
PlotLabel→Style[#,White],
```

```
ImageSize→Medium,ChartStyle→"MintColors",FrameStyle→Directive[White,12]]
&["Median"],
BoxWhiskerChart[{Table[RandomReal[],{i,0,50}],Table[RandomReal[],{i,0,50}],
Table[RandomReal[],{i,0,15}]},#,PlotLabel→Style[#,LightOrange],ImageSize→
Medium,ChartStyle→"MintColors",FrameStyle→Directive[Orange,12]]&["Basic"],
BoxWhiskerChart[{Table[RandomReal[],{i,0,50}],Table[RandomReal[],{i,0,50}],
Table[RandomReal[],{i,0,15}]},#,PlotLabel→Style[#,White],ImageSize→Medium,
ChartStyle→"MintColors",FrameStyle→Directive[White,12]]}&["Notched"],
{BoxWhiskerChart[{Table[RandomReal[],{i,0,50}],Table[RandomReal[],{i,0,50}],
Table[RandomReal[],{i,0,15}]},#,PlotLabel→Style[#,LightOrange],ImageSize→
Medium,ChartStyle→"MintColors",FrameStyle→Directive[Orange,12]]&["Outliers"],
BoxWhiskerChart[{Table[RandomReal[],{i,0,50}],Table[RandomReal[],{i,0,50}],
Table[RandomReal[],{i,0,15}]},#,PlotLabel→Style[#,White],ImageSize→Medium,
ChartStyle→"MintColors",FrameStyle→Directive[White,12]]&["Mean"],
BoxWhiskerChart[{Table[RandomReal[],{i,0,50}],Table[RandomReal[],{i,0,50}],
Table[RandomReal[],{i,0,15}]},#,PlotLabel→Style[#,LightOrange],ImageSize→
Medium,ChartStyle→"MintColors",FrameStyle→Directive[Orange,12]]&[
"Diamond"]}},FrameTicksStyle→18,Frame→{None,None,{{1,1}→True,{2,2}→True,
{1,3}→True}},FrameStyle→Directive[Thick,Red],Background→Black]
Out[48]=
```

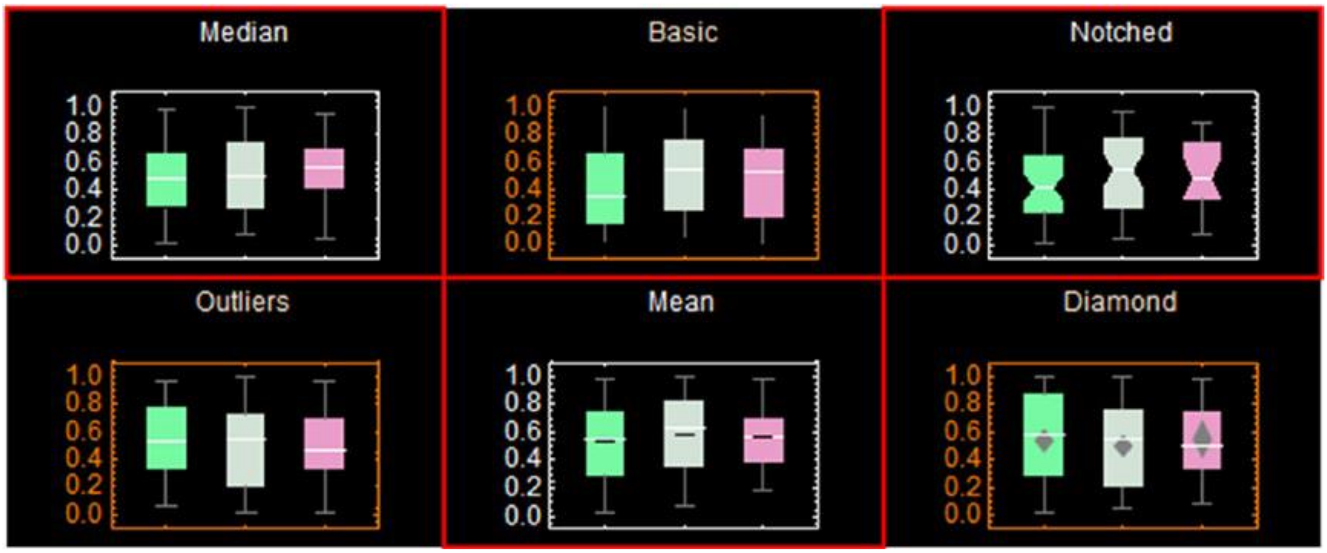


Рис. 3.12. Декілька коробчатих діаграм у вигляді масиву

Median – це специфікація за умовчанням; він показує медіану у центрі поля. Основне – показати лише коробку. Позначки показують довірчий інтервал для медіани. Викиди показують та позначають нетипові точки. Mean значення означає

середнє значення розподілу, а опція Diamond відзначає довірчий інтервал для середнього значення.

Діаграма скрипки використовується для візуалізації розподілу даних та густини ймовірності. Для побудови скрипкового графіка (Рис. 3.13) використовується команда DistributionChart.

```
In[49]:= DistributionChart[Table[i^Exp[i],{i,0,1,0.01}]]
Out[49]=
```

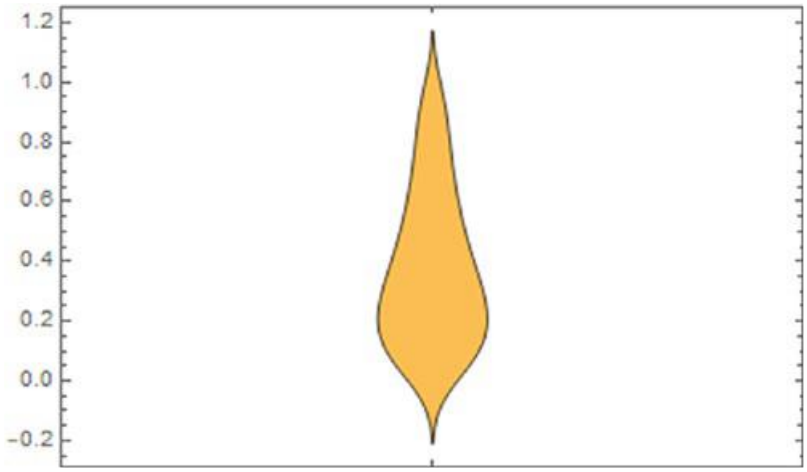


Рис. 3.13. Скрипковий графік

Графік, показаний на малюнку, є комбінацією прямокутного графіка і графіка щільності з кожної сторони, щоб показати, як розподіляються дані. DistributionChart має різні форми для побудови графіка.

```
In[50]:= GraphicsGrid[{{DistributionChart[Table[i^Exp[i],{i,0,2,0.1}],
ChartElementFunction→"SmoothDensity",PlotLabel→"SmoothDensity"],
DistributionChart[Table[i^Exp[i],{i,1,2,0.1}],ChartElementFunction→
"Density",PlotLabel→"Density",FrameStyle→Directive[Red,12]]},{
DistributionChart[Table[i^Exp[i],{i,0,1,0.09}],ChartElementFunction→
"HistogramDensity",PlotLabel→"HistogramDensity",FrameStyle→Directive[Red,
12]],DistributionChart[Table[i^Exp[i],{i,0,1,0.0112}],ChartElementFunction→
"PointDensity",PlotLabel→"PointDensity"]}},ImageSize→Medium,FrameStyle→
Directive[Thickness[0.02],LightGray],Dividers→{2→Directive[Black,Dotted],
2→Directive[Black,Dotted]},Frame→{1→False,False}]
Out[50]=
```

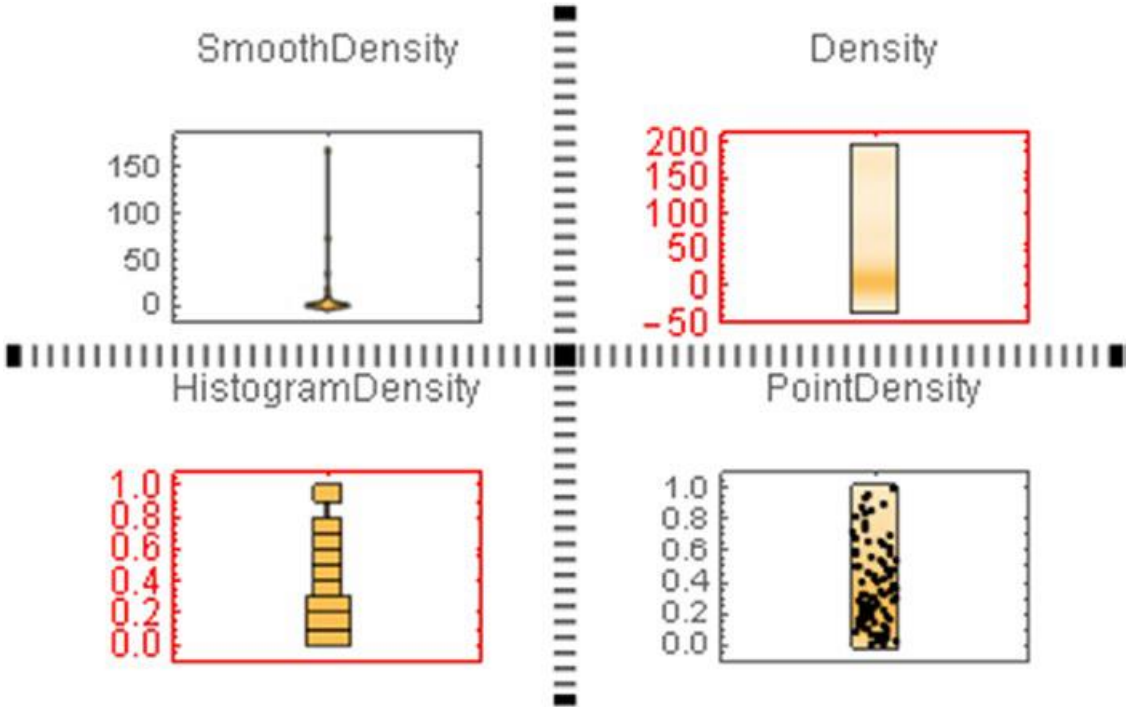


Рис. 3.14. Скрипкові діаграми різної форми

Інший спосіб додати параметри до діаграм — за допомогою панелі «Chart Element Schemes», яка знаходиться в меню «Palettes» (Palettes → Chart Element Schemes). Ця палітра показана Рис. 3.15.

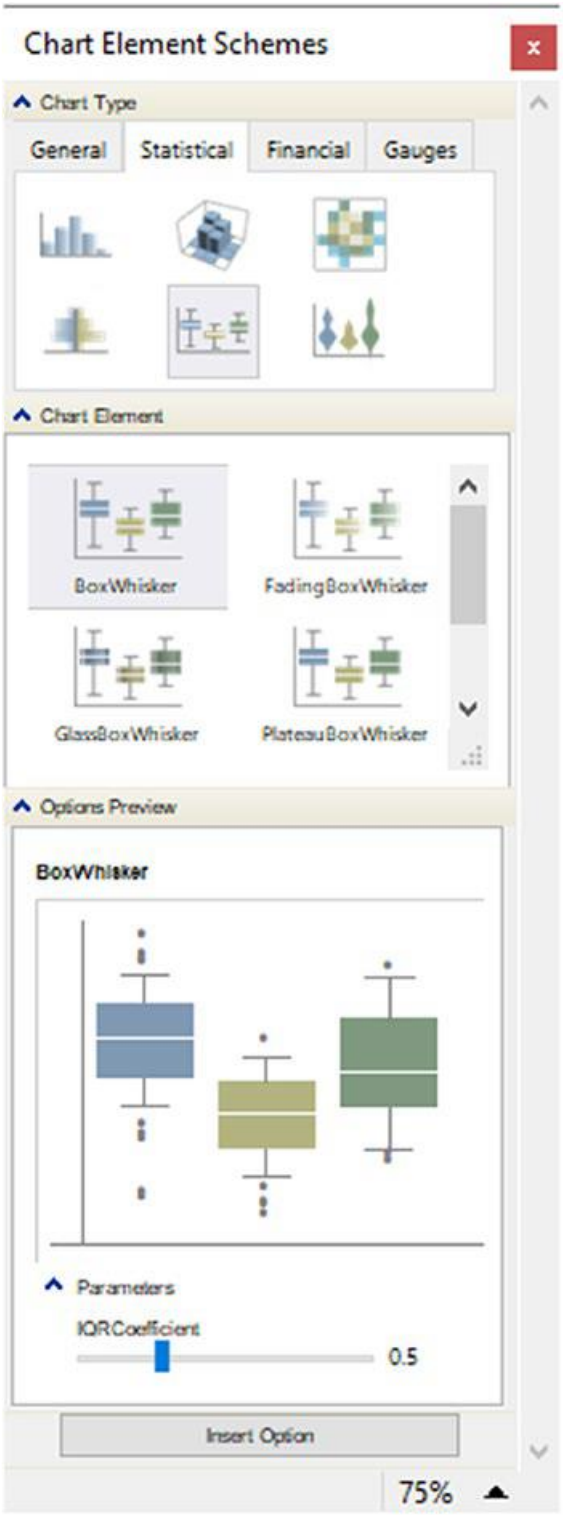


Рис. 3.16. Палітра Chart Element Schemes

На панелі наявні три категорії. Тип діаграми – тут ми вибираємо тип діаграми. Він містить чотири вкладки: (1) загальні, де знаходяться графіки із гістограм, секторів, нижнього колонтитулу та інших; (2) статистичні графіки, пов'язані з розподілом даних; (3) фінансовий, пов'язаний із діаграмами фінансових

даних; і (4) датчики, які є діаграми заходів. Друга категорія - вибір форми графіка за допомогою опції ChartElemenFunction. Третя категорія призначена для попереднього перегляду параметрів, вибраних із попередніх категорій. Щоб проілюструвати це, розглянемо наступні опції роботи. Спочатку ми побудуємо графік густини гістограми, а потім змінимо форму графіка за допомогою палітри. Щоб побудувати графік щільності гістограми, ми використовуємо команду DensityHistogram (Рис. 3.16).

```
In[51]:= DensityHistogram[Flatten[Table[{x^2+y^2,x^2-y^2},{x,0,2,0.1},{y,0,
2,0.1}],1],ChartBaseStyle→Red,ColorFunction→"SolarColors",Background→
Black,FrameStyle→Directive[White,Thick],FrameLabel→{"X","Y"},ImageSize→
300]
```

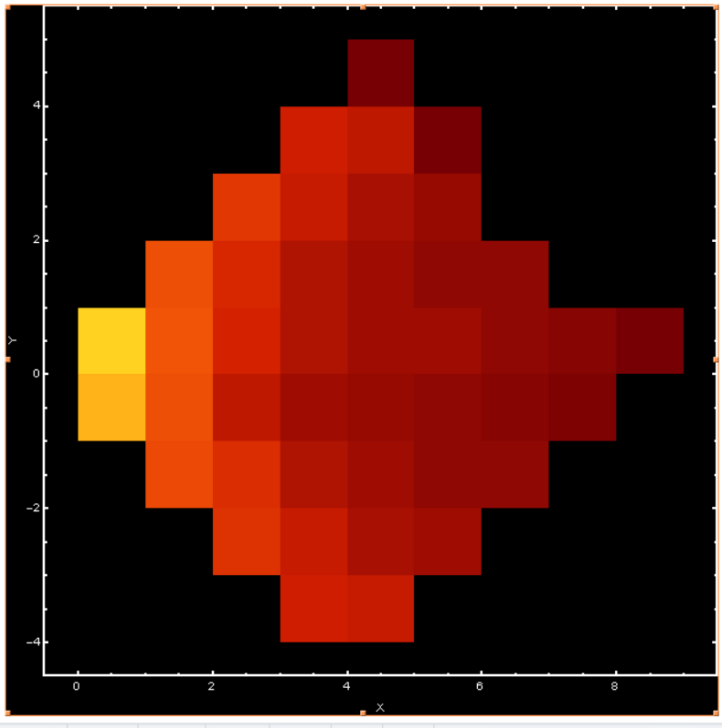


Рис. 3.16. Гістограма щільності

Як тільки графік буде готовий, ми додамо опцію із заголовком палітри та відкриємо палітру ChartElementSchemes. У типі діаграми ми клацаємо вкладку статистики та вибираємо діаграму DensityHistogram. Після того, як діаграма обрана, ми переходимо до елемента діаграми та вибираємо тип форми – Bubble. Потім ми переходимо до OptionsPreview, щоб подивитися, як виглядатиме наш графік; якщо ми натиснемо Shape, з'явиться спливаюче меню з іншими фігурами;

ми вибираємо шестикутник. На Рис. 3.17 показано, як має виглядати попередній перегляд вибраних елементів діаграми.

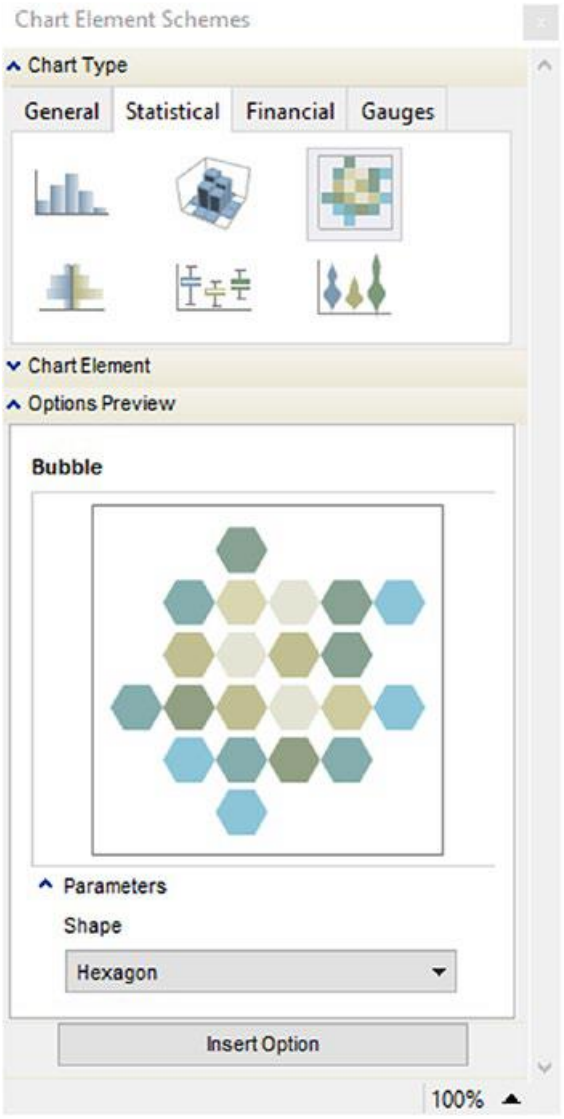


Рис. 3.17. Вибрані параметри гістограми щільності

Закінчивши вибір, натискаємо кнопку «Insert», щоб вона вставила наступний код: ChartElementFunction → ChartElementDataFunction [«Bubble», «Shape» → «Hexagon»]. Щоб правильно відобразити це, ми додаємо цей код як опцію та приступаємо до його побудови (Рис. 3.18), щоб побачити додану нову опцію.

```
In[52]:= DensityHistogram[Flatten[Table[{x^2+y^2,x^2-y^2},{x,0,2,0.1},{y,0,
2,0.1}],1],ChartBaseStyle→Red,ColorFunction→"SolarColors",Background→
Black,FrameStyle→Directive[White,Thick],FrameLabel→{"X","Y"},ImageSize→300,
ChartElementFunction→ChartElementDataFunction["Bubble","Shape"→"Hexagon"]]
Out[52]=
```

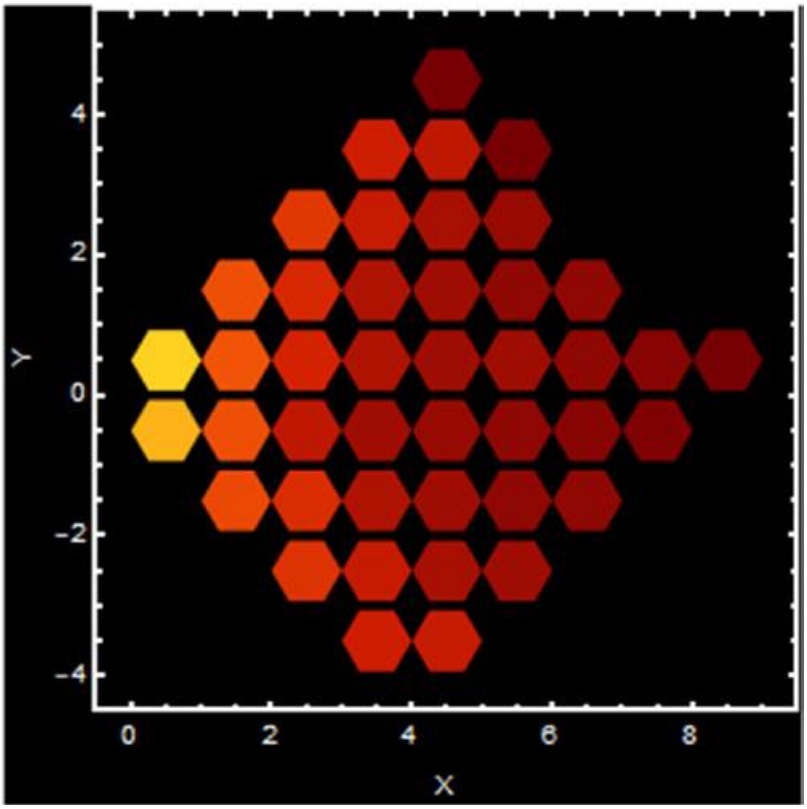


Рис. 3.18. Гістограма щільності шестикутника

Команда DensityHistogram дозволяє вибрати спосіб відображення розподілу даних по осях; це можуть бути вимірювання, коробчасті діаграми або гістограми, якщо ми виберемо тип методу як опцію.

```
In[53]:= Hist=Flatten[Table[{x^2+y^2,x^2-y^2},{x,0,2,0.1},{y,0,2,0.1}],1];
{MenuView[{DensityHistogram[Hist,Method→{"DistributionAxes"→True},
ColorFunction→GrayLevel,ChartBaseStyle→Directive[FaceForm[Opacity[0.5]],
EdgeForm[Red]],ChartLegends→BarLegend[Automatic,LegendMarkerSize→70],
PlotLabel→Style[" Density Histogram 1", Bold],ChartElementFunction→#,
ImageSize→200],
DensityHistogram[Hist,Method→{"DistributionAxes"→"Histogram"},ChartLegends→
BarLegend[Automatic,LegendMarkerSize→70],PlotLabel→Style[" Density
Histogram 2", Bold],ChartBaseStyle→EdgeForm[Thick],PlotTheme→"Scientific",
ChartElementFunction→#,ImageSize→200],
DensityHistogram[Hist,Method→{"DistributionAxes"→"BoxWhisker"},
ColorFunction→"BlueGreenYellow",PlotLabel→Style[" Density Histogram 3",
Bold],ChartLegends→BarLegend[Automatic,LegendMarkerSize→70],
ChartElementFunction→#,ImageSize→200]
}]&[ChartElementDataFunction["Bubble","Shape"→"Hexagon"]],{

GraphicsRow[{
DensityHistogram[Hist,Method→{"DistributionAxes"→True},ColorFunction→
GrayLevel,ChartBaseStyle→Directive[FaceForm[Opacity[0.5]],EdgeForm[Red]],
ChartLegends→BarLegend[Automatic,LegendMarkerSize→70],PlotLabel→Style["
Density Histogram 1", Bold],ChartElementFunction→#,ImageSize→130],
DensityHistogram[Hist,Method→{"DistributionAxes"→"Histogram"},ChartLegends→
BarLegend[Automatic,LegendMarkerSize→70],PlotLabel→Style[" Density
Histogram 2", Bold],ChartBaseStyle→EdgeForm[Thick],PlotTheme→"Scientific",
ChartElementFunction→#,ImageSize→130],
DensityHistogram[Hist,Method→{"DistributionAxes"→"BoxWhisker"},
ColorFunction→"BlueGreenYellow",PlotLabel→Style[" Density Histogram 3",
Bold],ChartLegends→BarLegend[Automatic,LegendMarkerSize→70],
ChartElementFunction→#,ImageSize→130]
}&[ChartElementDataFunction["Bubble","Shape"→"Hexagon"]]]}}
Out[53]=
```

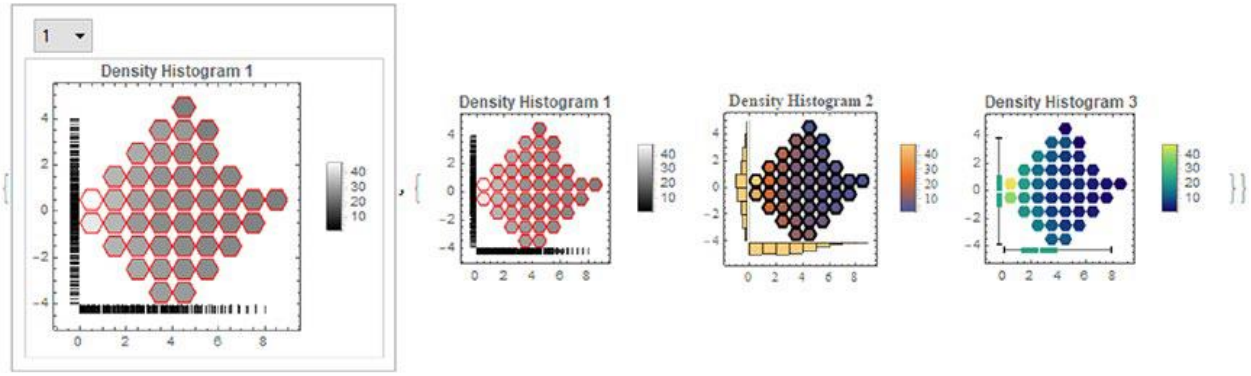


Рис. 3.19. Вигляд меню трьох різних графіків методу

Графіки відображаються у вигляді меню, тому для доступу до різних графіків вам необхідно вибрати кожен графік в меню. Зважаючи на це, ми подамо графіки в невеликому масштабі, щоб продемонструвати, як вони мають виглядати (Рис. 3.19). У першому графіку показані розміри розподілу даних з осях. Другий показує розподіл даних як гістограм, а третій — коробкоподібні діаграми.

3.4. Звичайний метод найменших квадратів та його реалізація мовою Wolfram Mathematica

Метод звичайних найменших квадратів переважно полягає у пошуку лінії, яка найкраще відповідає даним. Цей метод використовується для вивчення взаємозв'язку між залежною змінною та незалежною змінною. Метод заснований на вираженні пошуку лінії виду $y = mx + b$, де x - незалежна змінна, y - залежна змінна, m - нахил, а b - точка перетину з y. Розрахунок нахилу та сортування на початку координат b виходить з таких рівнянь.

$$m = \frac{n * \sum(x * y) - \sum x * \sum y}{n * \sum x^2 - \left|\sum x\right|^2}$$

$$b = \frac{\sum y * \sum x^2 - \sum x * \sum(x * y)}{n * \sum x^2 - \left|\sum x\right|^2}$$

Підсумовування позначається грецькою великою літерою сигма ($\sum$); n - обсяг даних у вибірці. Метод розраховується для пар виміряних даних і значень нахилу, а джерела перетину осі Y розраховуються для забезпечення найкращої відповідності даних лінії. Підставивши все у загальне рівняння, ми отримаємо рівняння лінії для набору даних. Щоб проілюструвати, на що схожий цей метод, розглянемо наступний приклад, використовуючи наші точки для залежної змінної та незалежної змінної.

```
In[54]:= Data={{-1,10},{0,9},{1,7},{2,5},{3,4},{4,3},{5,0},{7,-1}};
Grid[Transpose[Prepend[Data,{"X","Y"}]],Dividers→{2→True,2→True},
Alignment→Center]
Out[54]=
```

X	-1	0	1	2	3	4	5	7
Y	10	9	7	5	4	3	0	-1

Тепер нам потрібно обчислити дані, необхідні для отримання нахилу та перетину осі *y*.

```
In[55]:=
n=Length[Data];
SumX=Total@Data[[All,1]];
SumY=Total@Data[[All,2]];
SumXY=Total[Data[[All,1]]*Data[[All,2]]];
SumXSqre=Total@(Data[[All,1]]^2);
```

$$m = N@\frac{n*\text{SumXY}-\text{SumX}*\text{SumY}}{n*\text{SumXSqre}-\text{Abs}[\text{SumX}]^2};$$

$$b = N@\frac{\text{SumY}*\text{SumXSqre}-\text{SumX}*\text{SumXY}}{n*\text{SumXSqre}-\text{Abs}[\text{SumX}]^2};$$

Для вирішення рівняння форми $y=mx+b$ скористаємося командою Solve. Перший аргумент – це рівняння, а другий аргумент – змінна, яку потрібно розв’язати. Щоб ввести рівняння, ми повинні використовувати той самий подвійний запис, оскільки для інструкції установки використовується одинарне рівне.

```
In[56]:= Solve[SetPrecision[y==m*x+b,3],y]
Out[56]= {{y→8.47-1.47 x}}
```

В результаті рівняння лінії буде $y=1{,}47x+8{,}47$. Враховуючи це рівняння, ми побудуємо точки та лінію, яка найкраще відповідає цим точкам.

```
In[57]:=
Show[Plot[b+m x,{x,-1,8},PlotLegends→Placed[" Linear Fit: y=-1.47 x +
8.47",{0.6,0.8}],PlotRange→Automatic],ListPlot[Data,PlotStyle→Red]]
Out[57]=
```

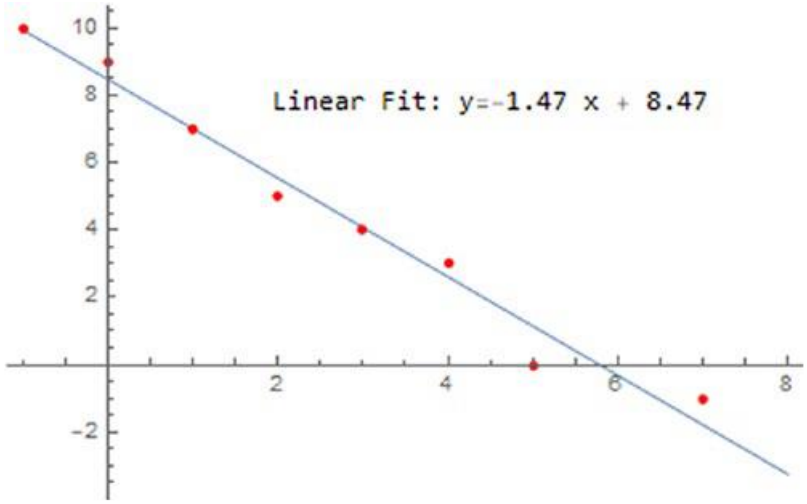


Рис. 3.20. Графік даних та апроксимована крива

Отримавши рівняння, бачимо, що це модель з негативним нахилом; це підтверджується графіком рівняння, показаним синім кольором.

Мірою, яка говорить нам про те, що обидві точки відповідають рівнянню, є коефіцієнт кореляції Пірсона, який називається *r*. Коли точки знайдені з позитивним нахилом, *r* матиме позитивне значення. Коли точки мають негативний нахил, *r* матиме негативне значення. Значення коефіцієнта визначає, наскільки правильне налаштування; це значення варіюється від -1 до 1. Коли значення *r* дорівнює 1 або -1, це говорить нам, що точки настроюються точно по лінії. Чим ближче до *r* -1 або 1, це вказує на наявність лінійної залежності між змінними дослідження. В іншому випадку, коли *r* дорівнює 0, це говорить нам про те, що налаштування неправильне, і, отже, можна дійти невтішного висновку, що очевидної лінійної залежності немає. Рівняння визначення коефіцієнта має такий вид:

$$r = \frac{\mathrm{cov}(x, y)}{\sigma_x \sigma_y},$$

де Cov є коваріацією *x*, *y*. Символи σ_x, σ_y позначають стандартні відхилення *x* та *y*.

```
In[58]:= r = N@(Covariance@@{Data[[All,1]],Data[[All,2]]}
          /(StandardDeviation@Data[[All,1]]*StandardDeviation@Data[[All,2]]))
Out[58]= -0.987814
```

Виданий нам результат близький до 1; ми можемо сказати, що пряма адекватно справедлива до даних. Хоча його можна обчислити за допомогою рівняння, у Mathematica є функція для цього розрахунку. Кореляція обчислює коефіцієнт із двох списків, тому нам потрібно ввести лише дані x в один список та дані з у в інший список.

```
In[59]:= N@Correlation[Data[[All,1]],Data[[All,2]]]
Out[59]= -0.987814
```

І отримуємо той самий результат, що й в попередньому випадку.

Навіть при використанні вищезгаданого процесу в системі Mathematica є функції, що спеціалізуються на пошуку кращої лінійної моделі за допомогою LinearModelFit. Враховуючи набір даних, ми пишемо команду LinearModelFit з даними для роботи та змінною для запису рівняння. Крім того, ми можемо вказати рівень точності налаштування за допомогою WorkPrecision.

```
In[60]:= Model=LinearModelFit[Data,x,x,WorkingPrecision→10]
Out[60]= FittedModel [8.473684211-1.466165414 x]
```

Як бачимо, ми отримуємо те саме рівняння, але з більшою точністю. Усередині моделі ми можемо отримати доступ до різних властивостей, пов'язаних з даними, моделлю та іншими параметрами коригування, а також засобами згоди, серед іншого. Щоб проілюструвати це, ми побачимо, як це зробити для опцій BestFit, BestFitParameters та Function, які мають повертати рівняння найкращої відповідності у вигляді списку, найкращих параметрів та побудови моделі для чистої функції відповідно. Дуже важливим аспектом є те, що спроба зробити прогноз щодо майбутнього значення з використанням підібраного рівняння (8,47–1,47*x*) із значеннями *x* поза діапазоном може призвести до появи ненормальних значень, оскільки ми насправді не встановили, чи є зв'язок рівняння поза діапазоном діапазон x фактично дотримується. На Рис. 3.21 показані розрахунки для апроксимованої кривої.

```
In[61]:= {"\n"Framed["Best Fit Parameters b and m: "<>ToString[Model[
"BestFitParameters" ]],Background→LightYellow],"\n"Framed["Equation:
"<>ToString[Model["BestFit" ]],Background→LightYellow],
"\n"Framed["Pure Function:"<>ToString[SetPrecision[Model["Function"],3]],
Background→LightYellow],"\n"Framed["r coeficcient:"<>ToString[r],
Background→LightYellow]}
Out[61]=
```

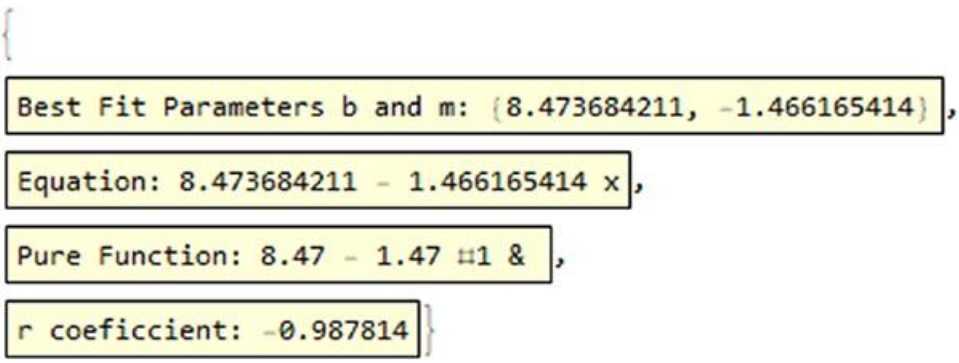


Рис. 3.21. Підгоночні параметри, рівняння та коефіцієнт Пірсона

Оскільки у нас є лінія, яка найкраще підходить, слід подумати, чи дійсно існує зв'язок між *x* і *y*. Тому природнім буде питання як дізнатися, чи правильно зроблене коригування описує лінійну залежність між змінними *x* та *y*. Для вирішення цієї проблеми існує концепція залишку.

Залишки можна використовувати як міру, щоб дізнатися, наскільки добре лінія відповідає досліджуваним точкам. Залишки є вертикальні відхилення, позитивні або негативні. Залишкова точка — це різниця між значенням залежної змінної і значенням, яке прогнозує коригування. Щоб отримати залишкові бали, ми пишемо властивість FitResiduals усередині моделі.

```
In[62]:= Model["FitResiduals"]
Out[62]= {0.06015038,0.52631579,-0.00751880,-0.54135338,-0.07518797,
0.39097744,-1.14285714,0.78947368}
```

За допомогою цих точок ми можемо отримати залишковий графік (Рис. 3.22), який є залежністю змінної *x* від залишкових точок.

```
In[63]:= ListPlot[Model["FitResiduals"],PlotStyle→{Red,Thick},PlotLabel→
"Residual Plot",AxesLabel→{Style["X",Bold],Style["residual points",
Bold]},Filling→Axis]
Out[63]=
```

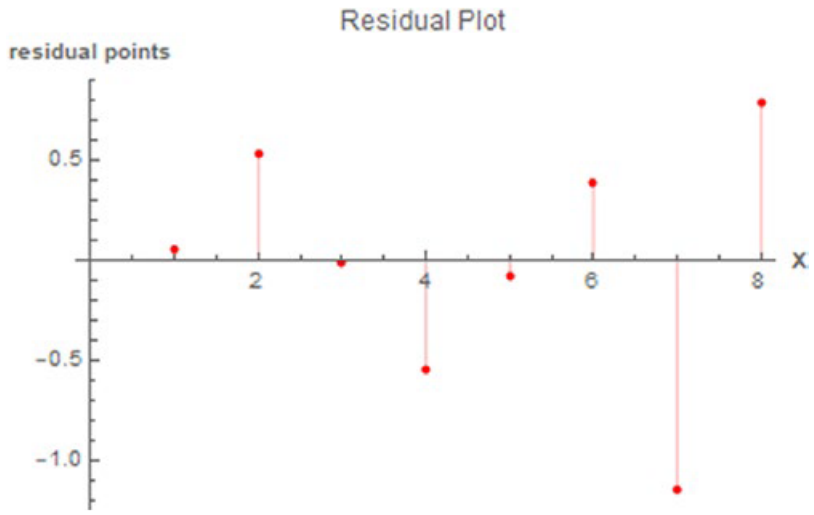


Рис. 3.22. Залишковий графік підібраних даних

Щоб відобразити лише спостережувані та попередні значення для одного прогнозу, використовуємо параметр SinglePredictionConfidenceIntervalTable.

```
In[64]:= Model["SinglePredictionConfidenceIntervalTable"]
Out[64]=
```

Observed	Predicted	Standard Error	Confidence Interval
10	9.93984962	0.78481739	{8.0194706,11.8602286}
9	8.47368421	0.74856412	{6.6420138,10.3053546}
7	7.00751880	0.72287410	{5.2387096,8.7763280}
5	5.54135338	0.70889670	{3.8067456,7.2759611}
4	4.07518797	0.70732661	{2.3444221,5.8059538}
3	2.60902256	0.71824519	{0.8515399,4.3665052}
0	1.14285714	0.74110068	{-0.6705509,2.9562652}
-1	-1.78947368	0.81811053	{-3.7913180,0.2123707}

Крім залишкових точок, ми можемо отримати таблицю з параметрів моделі, налаштованих за допомогою властивості ParameterTable.

```
In[65]:= Model["ParameterTable"]
Out[65]=
```

	Estimate	Standard Error	t-Statistic	P-Value
1	8.473684211	0.34167121	24.800697	2.8278226*10^-7
-x	-1.466165414	0.094310214	-15.5461996	4.4832546*10^-6

Розраховані коефіцієнти наведені в таблиці. Перший коефіцієнт є ординатою до початку координат, а коефіцієнт, пов’язаний зі змінною *x*, є нахилом. Два коефіцієнти мають відповідні стандартні помилки. Щоб дізнатися

довірчий інтервал для параметрів, ми пишемо властивість ParameterConfidenceIntervalTable.

```
In[66]:= Model["ParameterConfidenceIntervalTable"]
Out[66]=
```

	Estimate	Standard Error	Confidence Interval
1	8.473684211	0.34167121	{7.63764488,9.30972355}
x	-1.466165414	0.094310214	{-1.69693419,-.23539663}

Довірчий інтервал за замовчуванням становить 95%. Використовуючи ці довірчі значення, ми можемо побудувати точки, що знаходяться всередині або поза цим діапазоном (Рис. 3.23), отримуючи значення з прогнозів і встановлюючи для довірчого інтервалу значення 0,95.

```
In[67]:= Model[x];
Model["SinglePredictionBands",ConfidenceLevel→0.95];
Show[ListPlot[Data,PlotStyle→Red],
Plot[{Model[x],Model["SinglePredictionBands",ConfidenceLevel→0.95]},
{x,-1,10},Filling→{2→{1}}],PlotRange→{Automatic,{-1,10}},Frame→True,
ImageSize→400]
Out[67]=
```

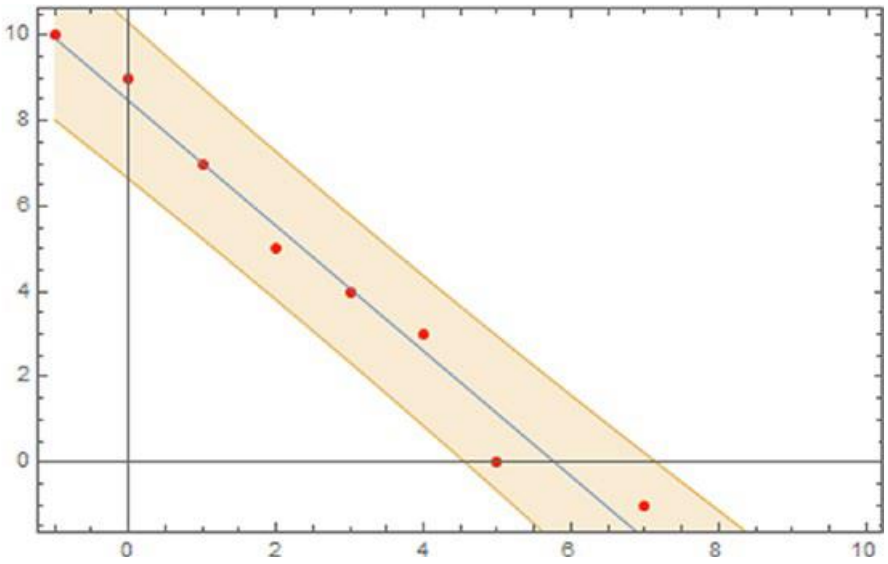


Рис. 3.23. Зафарбована область, що позначає довірчий інтервал 95%.

У загальному випадку, щоб отримати властивості, пов'язані із сумою квадратів помилок, ми використовуємо властивість ANOVATable.

```
In[68]:= Model["ANOVATable"]
Out[68]=
```

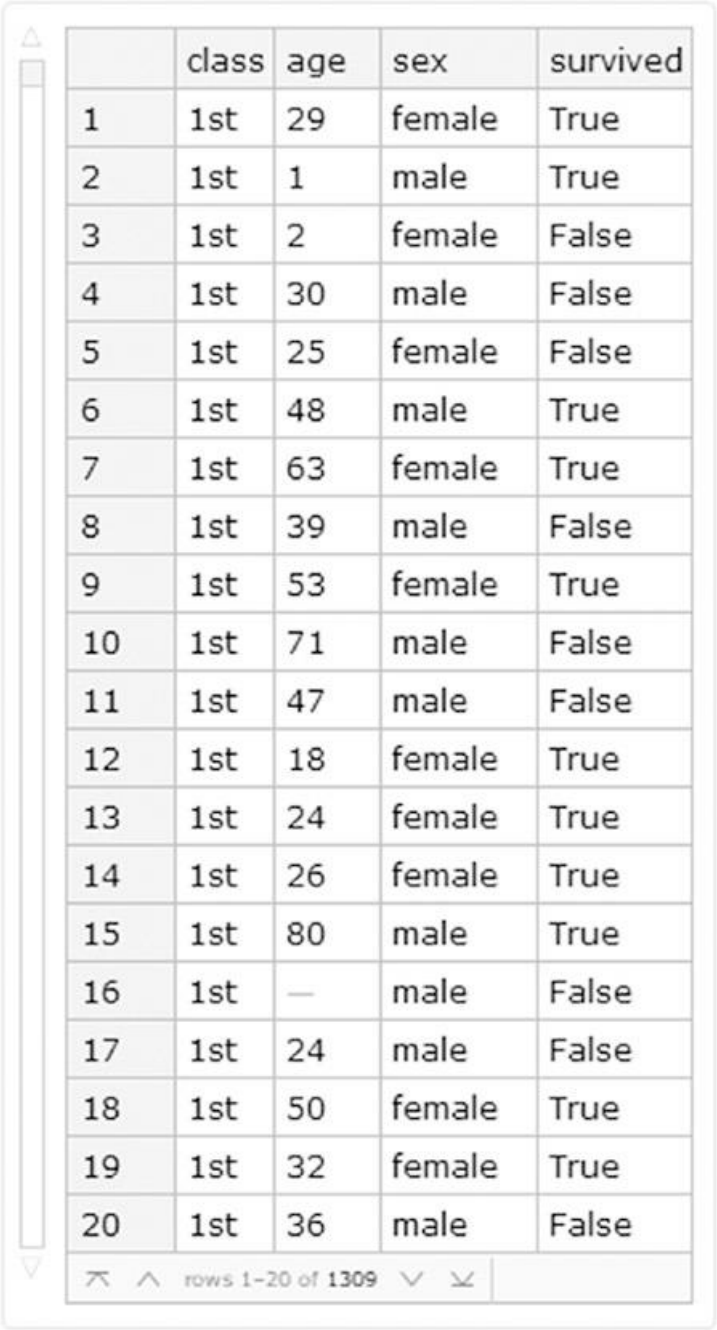
	DF	SS	MS	F-Statistic	P-Value
x	1	107.213346	107.213346	241.68432	4.48325*10^-6
Error	6	2.6616541	0.44360902		
Total	7	109.8750000			

3.5. Поняття про логістичну регресію та її реалізацію мовою Wolfram. Набір даних Titanic

Логістична регресія - це метод, який зазвичай використовується в статистиці, але він також використовується в машинному навчанні. Логістична регресія працює, враховуючи, що значення змінної відповіді приймають лише два значення: 0 та 1; це також можна інтерпретувати як хибну чи справжню умову. Це двійковий класифікатор, який використовує функцію для прогнозування ймовірності виконання умови, залежно від того, як побудована модель. Зазвичай цей тип моделі використовується для класифікації, оскільки він здатний надати нам ймовірності та класифікації, оскільки значення логістичної регресії коливаються між двома значеннями. У логістичній регресії цільової змінної є двійкова змінна, що містить закодовані дані.

У наступному прикладі ми будемо використовувати набір даних Titanic, який визначає статус виживання пасажирів. Як змінні використовуються клас, вік, стать та умови виживання. Ми завантажимо дані у вигляді набору даних (Рис. 3.24 з SampleData і перерахуємо рядки набору даних.

```
In[1]:= Titanic=Query[AssociationThread[Range[Length@#]→Range[Length@#]]]
[ExampleData[{"Dataset","Titanic"}]]&[ExampleData[{"Dataset","Titanic"}]]
Out[1]=
```



	class	age	sex	survived
1	1st	29	female	True
2	1st	1	male	True
3	1st	2	female	False
4	1st	30	male	False
5	1st	25	female	False
6	1st	48	male	True
7	1st	63	female	True
8	1st	39	male	False
9	1st	53	female	True
10	1st	71	male	False
11	1st	47	male	False
12	1st	18	female	True
13	1st	24	female	True
14	1st	26	female	True
15	1st	80	male	True
16	1st	—	male	False
17	1st	24	male	False
18	1st	50	female	True
19	1st	32	female	True
20	1st	36	male	False

Рис. 3.24. Набір даних Titanic

Погляньмо на розміри даних за допомогою команди Dimensions.

```
In[2]:= Dimensions@Titanic
Out[2]= {1309,4}
```

Інтерпретуючи результат, бачимо, що набір даних складається з 1309 рядків по 4 стовпці. У наборі даних є чотири стовпці, класифіковані за класом, віком, статтю та статусом того, хто вижив. Якщо ми скористаємося пробілом, ми побачимо, що є деякі елементи, які не реєструють введення даних. Щоб дізнатися,

які стовпці містять відсутні дані, виконайте наступний код, підрахувавши кількість елементів, що відповідають шаблону Missing, у кожному зі стовпців.

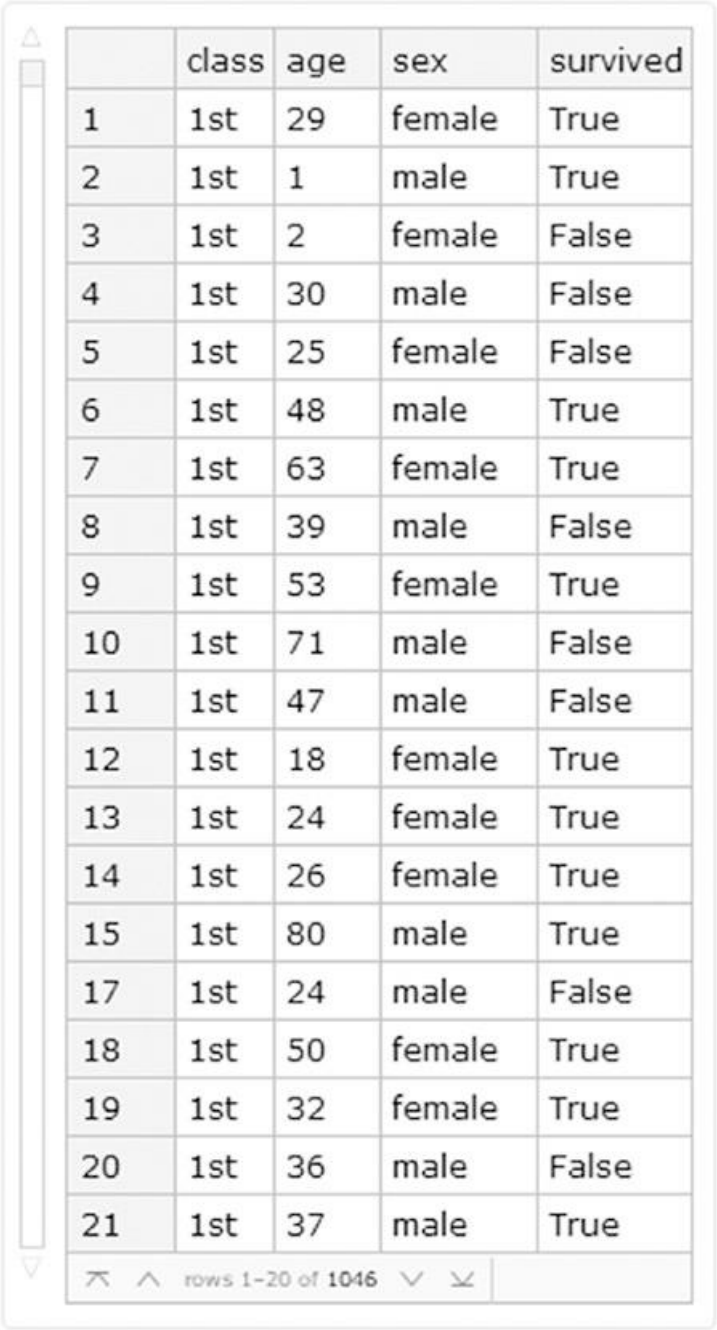
```
In[3]:= Query[Count[_Missing],#]@Titanic&/@{"class","age","sex","survived"}
Out[3]= {0,263,0,0}
```

В результаті ми отримуємо 263 пропущені значення в стовпці віку і нуль в інших. Давайте видалимо рядки, що містять ці дані, але спочатку ми витягнемо номери рядків з даних, вибравши елементи зі стовпця віку, рівні Missing, а потім витягуючи ідентифікатори рядків.

```
In[4]:= Query[Select[#age==Missing[]&]][Titanic];
Normal@Keys@%
Out[5]= {16,38,41,47,60,70,71,75,81,107,108,109,119,122,126,135,148,153,158
,167,177,180,185,197,205,220,224,236,238,242,255,257,270,278,284,294,298,31
9,321,36,4,383,385,411,470,474,478,484,492,496,525,529,532,582,596,598,673,
681,682,683,706,707,757,758,768,769,776,790,796,799,801,802,803,805,806,809
,813,814,816,817,820,836,843,844,853,855,857,859,866,872,873,875,877,880,88
3,887,888,901,902,903,904,919,921,922,923,924,927,928,929,930,931,932,941,9
43,945,946,947,949,955,956,957,958,959,962,963,972,974,977,983,984,985,988,
989,990,992,994,995,998,999,1000,1001,1002,1003,1004,1005,1006,1007,1010,10
13,1014,1015,1017,1019,1023,1024,1028,1029,1030,1031,1033,1034,1035,1036,10
37,1038,1039,1040,1042,1043,1044,1045,1053,1054,1055,1056,1070,1071,1072,10
73,1074,1075,1077,1078,1079,1081,1082,1086,1096,1110,1115,1116,1117,1122,11
23,1124,1125,1129,1133,1136,1137,1138,1139,1150,1151,1152,1155,1156,1160,11
63,1164,1165,1167,1168,1169,1171,1173,1174,1175,1176,1177,1178,1179,1180,11
81,1185,1186,1187,1194,1195,1196,1198,1199,1200,1201,1203,1213,1214,1215,12
16,1217,1220,1222,1242,1243,1244,1246,1247,1248,1250,1251,1254,1256,1263,12
69,1283,1284,1285,1292,1293,1294,1298,1303,1304,1306}
```

Ці числа є рядками, що містять дані для стовпця віку. Щоб їх усунути, ми використовуємо команду DeleteMissing з огляду на те, що на рівні 1 відсутні дані. Остаточний набір даних показаний на Рис. 3.25.

```
In[5]:= Titanic=DeleteMissing[Titanic,1,1]
Out[5]=
```



	class	age	sex	survived
1	1st	29	female	True
2	1st	1	male	True
3	1st	2	female	False
4	1st	30	male	False
5	1st	25	female	False
6	1st	48	male	True
7	1st	63	female	True
8	1st	39	male	False
9	1st	53	female	True
10	1st	71	male	False
11	1st	47	male	False
12	1st	18	female	True
13	1st	24	female	True
14	1st	26	female	True
15	1st	80	male	True
17	1st	24	male	False
18	1st	50	female	True
19	1st	32	female	True
20	1st	36	male	False
21	1st	37	male	True

Рис. 3.25. Набір даних Titanic без пропущених значень

Щоб переконатися, що пропущених даних більше немає, можна застосувати той же код до підрахунків або, наприклад, переглянувши ключі віддалених рядків.

```
In[6]:= Titanic[Key[16]]
Out[6]= Missing[KeyAbsent,16]
```

Це означає, що з ключем 16 немає вмісту. При бажанні можна перевірити всі ключі, використовуючи список рядків відсутніх даних.

Видаляючи дані, ми можемо підрахувати кількість елементів, що складаються з кожного класу, статі та статусу виживання.

```
In[7]:= Dataset@
<|
"Class"→Query[Counts,"class"]@Titanic,"Sex"→ Query[Counts,"sex"]@Titanic,
"Survival status"→Query[Counts,"survived"]@Titanic
|>
Out[7]=
```

Class	1st 284 2nd 261 3rd 501
Sex	female 388 male 658
Survival status	True 427 False 619

Рис. 3.26. Базові елементи, що враховуються за класом, статтю та статусом виживання

Виключивши рядки з відсутніми елементами, бачимо, що набір даних складається з 284 елементів першого класу, 261 елемента другого класу і 501 елемента третього класу (Рис. 3.26). Також слід звернути увагу, що більше половини зареєстрованих пасажирів були чоловіками і що загиблих було більше, ніж тих, що вижили. Це можна перевірити графічно, показавши відсотки (Рис. 3.27). Той самий підхід застосовується до стовпців класу та статі.

```
In[8]:= Row[{PieChart[{N@(#[[1]]/Total@#),N@(#[[2]]/Total@#)}&[Counts
[Query[All,"survived"][Titanic]]], PlotLabel→Style["Percentage of
survival",#3,#4], ChartLegends→ {"Survived", "Died"}, ImageSize→#1,ChartS
tyle→#2,LabelingFunction→(Placed[Row[{SetPrecision[100#,3],"%"}],"RadialC
allout"]&)],
PieChart[{N@(#[[1]]/Total@#),N@(#[[2]]/Total@#)}&[Counts[Query[All,
"sex"][Titanic]]], PlotLabel→Style["Percentage by sex",#3,#4],
ChartLegends→{"Female", "Male"}, ImageSize→#1,ChartStyle→#2,LabelingFunc
tion→(Placed[Row[{SetPrecision[100#,3],"%"}],"RadialCallout"]&)],
PieChart[{N@(#[[1]]/Total@#),N@(#[[2]]/Total@#),N@(#[[3]]/Total@#)}&[Counts
[Query[All,"class"][Titanic]]], PlotLabel→Style["Percentage by class",#3,#4],
ChartLegends→{"1st", "2nd","3rd"}, ImageSize→#1,ChartStyle→#2,Labeling
Function→(Placed[Row[{SetPrecision[100#,3],"%"}],"RadialCallout"]&)]},"----
"]&[200,{ColorData[97,20],ColorData[97,13],ColorData[97,32]},Black,20]
Out[8]=
```

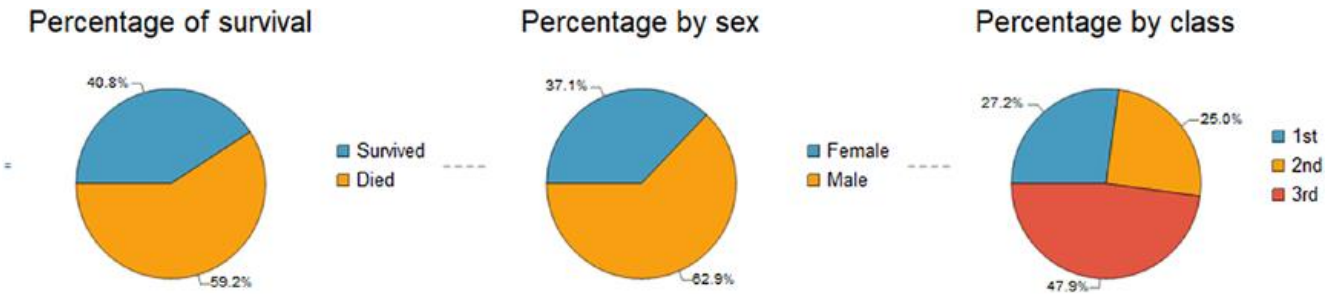


Рис. 3.27. Кругові діаграми класу, статі та статусу виживання

У цьому випадку ми збираємось передбачити виживання пасажирів Титаніка. Ми побудуємо модель, яка визначатиме, чи виживе даний клас, вік та стать чи ні. Характеристики будуть класом, віком та статтю, а метою буде статус виживання. Ми збираємося використовувати ці змінні як ознаки, які потім модель буде використовувати для класифікації того, чи виживуть їх клас, вік і стать чи ні, що є нашою цільовою змінною. Для цього ми ділимо набір даних на 80% навчальних (837 елементів) та 20% тестових (209 елементів). Щоб розділити набір даних, спочатку виконаємо випадкову вибірку; після цього ми вилучимо ключі ідентифікаторів і створимо новий набір даних, розділений на навчальний та тестовий набори (Рис. 3.28).

```
In[9]:= BlockRandom[SeedRandom[8888];
RandomSample[Titanic]];
Keys@Normal@Query[All][%];
{train,test}={%[[1;;837]],%[[838;;1046]]};
dataset=Query[<|"Train"→{Map[Key,train]},"Test"→{Map[Key,test]} |> ]
[Titanic]
Out[9]=
```

		class	age	sex	survived
Train	410	2nd	36	male	False
	537	2nd	32	female	True
	874	3rd	42	male	False
	691	3rd	22	male	False
	1021	3rd	21	male	False
	852	3rd	45	female	True
	705	3rd	21	male	False
	743	3rd	45	male	True
	515	2nd	2	male	True
	658	3rd	1	female	True
	837 total ›				
Test	1227	3rd	19	male	False
	188	1st	16	female	True
	397	2nd	34	female	True
	944	3rd	37	female	False
	262	1st	35	male	True
	95	1st	4	male	True
	1080	3rd	22	female	True
	918	3rd	39	male	True
	517	2nd	37	male	False
	425	2nd	30	male	False
	209 total ›				

Рис. 3.29. Набір даних Titanic, розділений на навчальний та тестовий набори

3.6. Застосування функції класифікації даних

Команда Classify - це ще одна суперфункція, яка використовується у схемі машинного навчання Wolfram Language. Цю функцію можна використовувати у задачах, що складаються з розв'язання задачі класифікації. Дані, які приймає ця функція, є числові, текстові, звукові та графічні дані. Вхідні дані цієї функції можуть бути такими ж, як і функції Predict{x → y}. Однак також можна вводити дані у вигляді списку елементів, асоціації елементів або набору даних. І тут представимо його як набір даних. У цьому випадку ми вийдемо дані з формату набору даних, вказавши, що вхідні стовпці (клас, вік, стать) вказують на ціль (збереглися). Тепер ми створимо функцію класифікатора (Рис. 3.30) з такими параметрами: Method → {LogisticRegrade, L1 → Automatic_, L2 → Automatic}. При виборі Automatic_ ми дозволяємо Mathematica вибрати найкращу комбінацію параметрів L1 та L2. Для OptimizationMethod встановіть метод StochasticGradientDescent. А для мети продуктивності встановіть якість. Нарешті, вибираємо початкове значення зі значенням 100 000 та процесорний модуль як цільовий пристрій. Методами оптимізації логістичної регресії є алгоритм Бройдена-Флетчера-Гольдфарба-Шенно з обмеженою пам'яттю (LBFGS), StochasticGradientDescent (метод стохастичного градієнта) та Newton (метод Ньютона). Вони призначені з метою оцінки параметрів логістичної функції. Побудова правил буде виконуватися на основі даних усередині набору даних із використанням мови запитів.

```
In[10]:= CF=Classify[Flatten[Values[Normal[Query["Train",All,All,{#class,#a
ge,#sex}→ #survived&][dataset]]]],Method→{"LogisticRegression","L1Regular
ization"→ Automatic,"L2Regularization"→ Automatic,"OptimizationMethod"→"
StochasticGradientDescent"},PerformanceGoal→"Quality",RandomSeeding→10000
0,TargetDevice→"CPU",TrainingProgressReporting→None]
Out[10]=
```

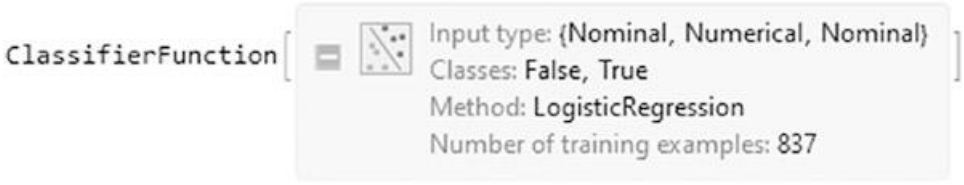


Рис. 3.31. Об’єкт ClassifierFunction для масиву даних

Після навчання, як у випадку з функцією Predict, функція Classify повертає об'єкт функції класифікатора (Рис. 3.31) замість функції прогнозування. Перевіряючи функцію класифікатора, ми бачимо два типи вхідних даних: номінальний та числовий. Класи, які є статусом виживання - хибним чи істинним. Метод, що використовується (логістична регресія); та кількість прикладів (837). Для отримання інформації про модель використовуємо команду Information. Розглянемо звіт роботи моделі (Рис. 3.32).

```
In[11]:= Information[CF]
Out[11]=
```

Якщо клацнути стрілки над графіками, відобразяться три графіки: крива навчання, точність та крива навчання для всього алгоритму. Якщо навести вказівник миші на останній рядок, з'явиться підказка з відповідними параметрами і використаним методом.

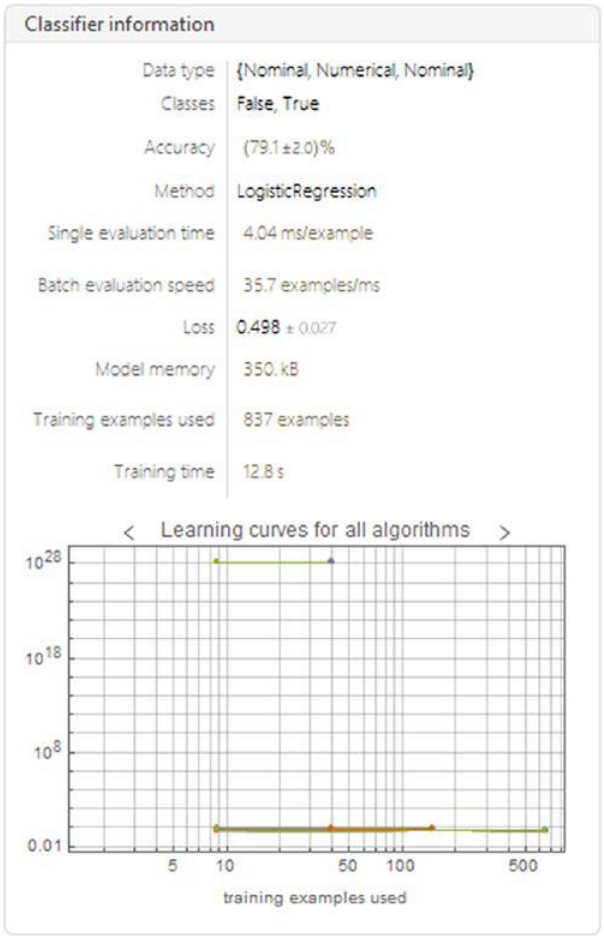


Рис. 3.32. Інформація про навчену функцію класифікатора

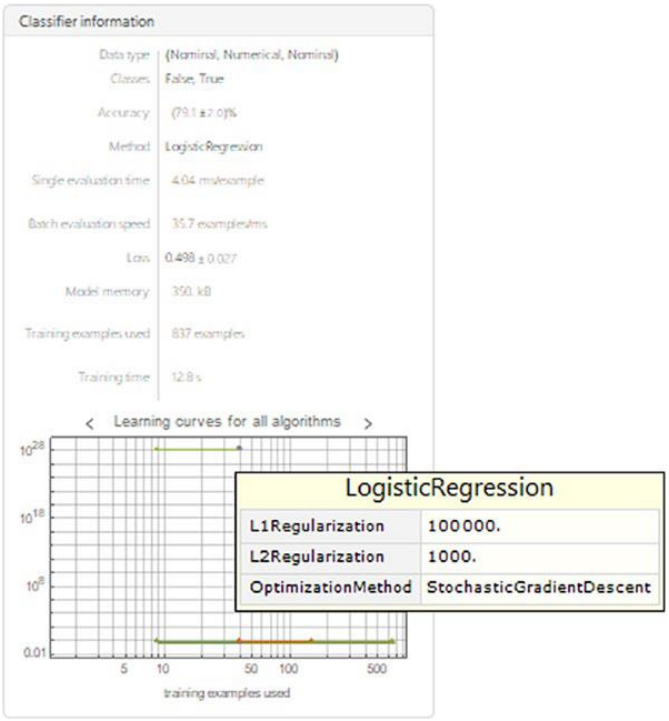


Рис. 3.33. Підказка щодо специфікацій алгоритму з методу логістичної регресії

Ми бачимо, що точність моделі становить приблизно 78%. Клацаючи стрілки на графіках, ми також спостерігаємо, що крива навчання та точність не показують жодних ознак покращення порівняно з 500 і більше прикладами. Щоб отримати доступ до всіх властивостей навченої моделі, додаємо параметр «Properties» у розділі «Information».

```
In[12]:= Information[CF,"Properties"]
Out[12]= {Accuracy,BatchEvaluationSpeed,BatchEvaluationTime,Classes,
ClassNumber,ClassPriors,EvaluationTime,ExampleNumber,FeatureNames,
FeatureNumber,FeatureTypes,FunctionMemory,FunctionProperties,
IndeterminateThreshold,L1Regularization,L2Regularization,LearningCurve,
MaxTrainingMemory,MeanCrossEntropy,Method,MethodDescription,MethodOption,
OptimizationMethod,PerformanceGoal,Properties,TrainingClassPriors,
TrainingTime,UtilityFunction}
```

Тепер подивимося, яка можливість даних: клас = 3-й, вік = 23 і стать = чоловічий. Probability → ім'я або номер класу або TopProbabilities → кількість найімовірніших класів.

```
In[13]:= CF[{"3rd",23,"male"},{"Probability"→ False,"TopProbabilities"→ 2}]
Out[13]= {0.839494,{False→0.839494,True→0.160506}}
```

Імовірності з останнього прикладу показують, що статус виживання пасажира може бути більш схильний до False. Щоб переглянути повні властивості нової класифікації, введемо приклад, а потім Properties. Включені властивості: Decision (найкращий вибір класу відповідно до ймовірностей та його функції корисності) і Distribution (об'єкт категоріального розподілу). Імовірності кожного класу відображаються у вигляді асоціацій: ExpectedUtilities (очікувані ймовірності), LogProbabilities (ймовірності натурального логарифму), Probabilities (ймовірності всіх класів) та TopProbabilities (найбільш ймовірний клас). Це відображається у наступному наборі даних.

```
In[14]:= Dataset@
AssociationMap[CF[{"3rd",23,"male"},#] &,{"Decision","Distribution",
"ExpectedUtilities","LogProbabilities","Probabilities","TopProbabilities"}]
Out[14]=
```

Decision	False
Distribution	CategoricalDistribution[Input type: Scalar Categories: False True]
ExpectedUtilities	⟨|False → 0.839494, True → 0.160506, Indeterminate → 0.|⟩
LogProbabilities	⟨|False → -0.174956, True → -1.82943|⟩
Probabilities	⟨|False → 0.839494, True → 0.160506|⟩
TopProbabilities	{False → 0.839494, True → 0.160506}

Рис. 3.34. Властивості функції класифікатора навченої моделі

3.7. Тестування моделі

Тепер ми протестуємо модель на тестових даних за допомогою команди ClassifierMeasurements (Рис. 3.35), додавши функцію та тестовий набір як аргументи та обчисливши невизначеність.

```
In[15]:= CM=ClassifierMeasurements[CF,Flatten[Values[Normal[Query["Test",
All,All,{#class,#age,#sex}→ #survived&][dataset]]]],ComputeUncertainty→
True,RandomSeeding→8888]
Out[15]=
```

ClassifierMeasurementsObject[Classifier: LogisticRegression
Number of test examples: 209
Number of classes: 2
Accuracy: 0.737 ±0.031]

Рис. 3.35. Об'єкт ClassifierMeasurements функції класифікатора

Об'єкт, що повертається, називається ClassifierMeasurementsObject, який використовується для пошуку властивостей ClassifierFunction після тестування набору тестів. Тепер подивимося звіт:

```
In[16]:= CM["Report"]
Out[16]=
```

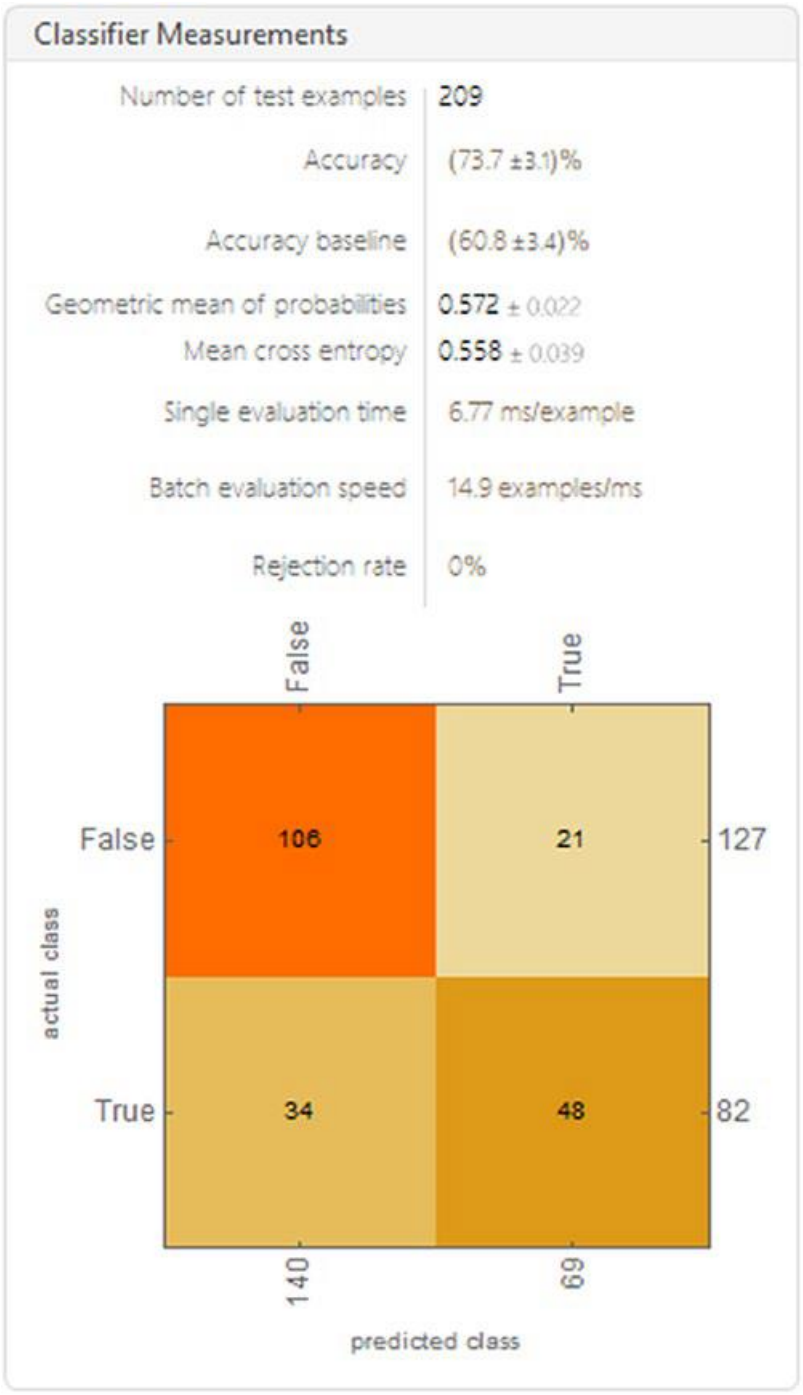


Рис. 3.36. Звіт про тестовану модель

Звіт, показаний малюнку, містить, серед іншого, таку інформацію, як кількість тестових прикладів, точність і базовий рівень точності. Він також показує нам матрицю плутанини, яка показує результати прогнозування моделі класифікації, показуючи кількість правильних і неправильних прогнозів; в даному випадку вони розбиваються по класах і повертають або false, або true, що дає нам уявлення про помилки, які допускає модель, і про тип помилок, які вона робить.

По суті, він показує нам справжні позитивні та справжні негативні, хибно позитивні та хибно негативні результати для кожного класу.

Тепер ми подивимося на графік (матрицю неточностей) безпосередньо.

```
In[17]:= CM["ConfusionMatrixPlot"]
Out[17]=
```

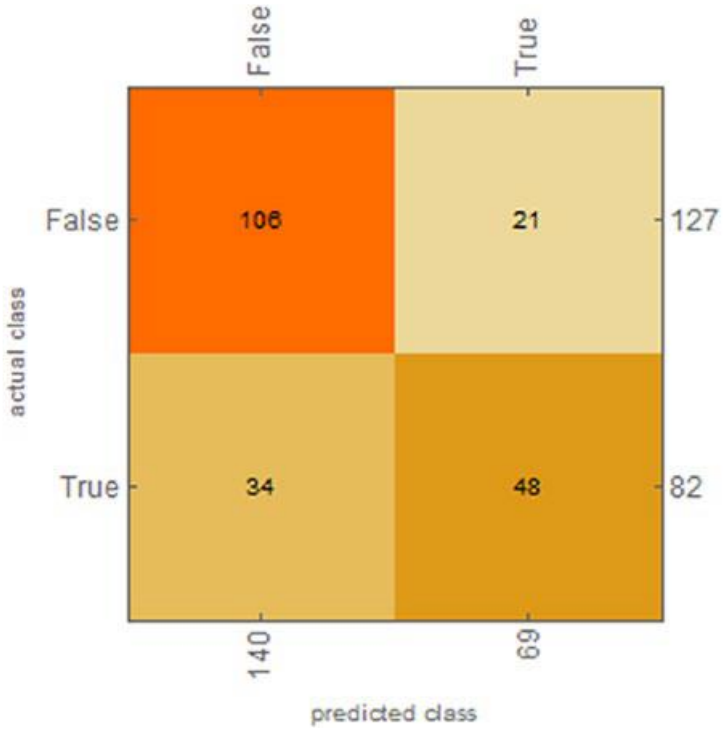


Рис. 3.37. Графік матриці неточностей моделі, що тестується.

Щоб отримати значення матриці неточності, використаємо модуль CM[“ConfusionMatrix”] або клас CM[“ConfusionFunction”]. Дивлячись на графік, ми бачимо, що модель класифікувала, починаючи зліва направо вгору, 106 прикладів хибних значень, які були правильно класифіковані, 21 приклад хибних значень як дійсні, 34 приклади правильних значень як хибних і 48 прикладів дійсних значень правильно. Щоб краще візуалізувати продуктивність, погляньмо на криві ROC (Рис. 3.38) для кожного класу, їх відповідні значення, а також коефіцієнт кореляції Метьюза і значення AUC.

```
In[18]:= {CM["ROCCurve"],Dataset@<|{"AUC"→CM["AreaUnderROCCurve"]},
{"MCC"→CM["MatthewsCorrelationCoefficient"]}|>}
```

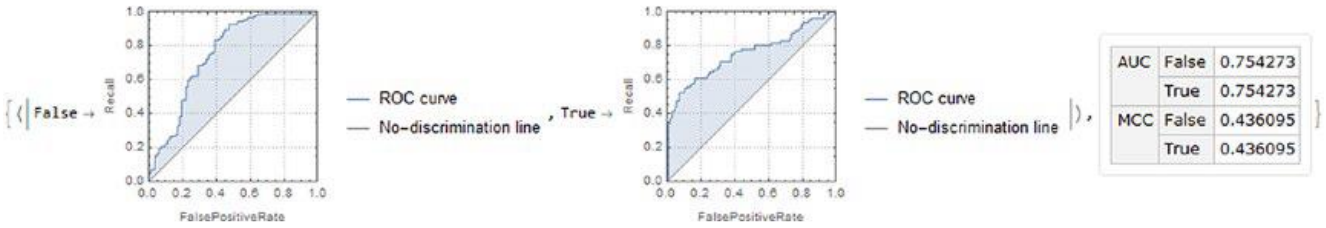


Рис. 3.38. Криві ROC для кожного класу разом зі значеннями AUC та MCC.

Очевидно, що обидва класи мають однакові значення, але, порівняно з кривою ROC, ми бачимо, що клас False мав кращу класифікацію, ніж клас True; давайте подивимося на найменш визначені приклади, щоб побачити, що клас True має гірші певні приклади, ніж у False. Завдяки цьому ми можемо показати менш точні результати моделі, які мають найвищий розподіл ентропії та середню перехресну ентропію для кожного класу.

```
In[19]:= CM[{"LeastCertainExamples","ClassMeanCrossEntropy"}]
Out[19]= {{{3rd,39,female}→False,{3rd,38,female}→True,{3rd,37,female}→
False,{3rd,37,female}→False,{3rd,36,female}→True,{3rd,32,female}→False,
{1st,4,male}→True,{3rd,30,female}→False,{3rd,28,female}→False,{3rd,27,
female}→True},<|False→0.363541,True→0.85931|>}
```

Щоб отримати значення коефіцієнта MCC, використаємо наступні властивості: FalseDiscoveryRate, FalsePositiveRate, FalseNegativeRate (частота хибно позитивних і хибно негативних відкриттів для кожного класу), FalseNegativeExamples, FalseNegativeNumber (істинно негативні значення), FalsePositiveExamples і FalsePositiveExamples і FalsePositiveExamples. Далі вони показані у короткій формі.

```
In[20]:= CM[#]&/@{"FalseDiscoveryRate","FalseNegativeRate",
"FalsePositiveRate"}
Out[20]= {<|False→0.242857,True→0.304348|>,<|False→0.165354,
True→0.414634|>,<|False→0.414634,True→0.165354|>}
```

Інший спосіб перевірити, чи модель стабільно поводиться в прогнозах, — це переглянути значення ключових показників, таких як точність, відгук, точність F1Score і графік відхилення точності (Рис. 3.39). Розглянемо ці показники для моделі.

```
In[21]:= CM[{"Accuracy","Recall","F1Score","Precision",
"AccuracyRejectionPlot"}]//TableForm
Out[21]//TableForm=
```

```
0.736842
⟨|False → 0.834646, True → 0.585366|⟩
⟨|False → 0.794007, True → 0.635762|⟩
⟨|False → 0.757143, True → 0.695652|⟩
```

Рис. 3.39. Таблична форма для значень Accuracy, Recall, F1 Score, Precision та Accuracy Rejection Plot.

Щоб переглянути пов'язані показники точності, введіть такі властивості: Accuracy (кількість правильно класифікованих прикладів), AccuracyBaseline (точність прогнозування загального класу) та AccuracyRejectionPlot (графік ARC, крива відхилення точності). Однак, щоб знайти інформацію про ймовірність та прогнозований клас тестового набору, використовуйте такі властивості: DecisionUtilities (значення функції корисності для кожного прикладу в тестовому наборі), Probabilities (ймовірності для кожного прикладу в тестовому наборі) та ProbabilityHistogram (гістограма ймовірностей класів). Давайте подивимося, як поводиться можливість, побудувавши графік ймовірності статусу виживання пасажира (Рис. 4.16). Пам'ятайте, що стан False означає, що пасажир не вижив, а стан True означає, що пасажир вижив.

```
In[22]:= TruPlot=
{Plot[{CF[{#1,age,#4},"Probability"→ #6 ],CF[{#2,age,#4},"Probability"→
#6 ],CF[{#3,age,#4},"Probability"→ #6 ]}, {age,0,90},PlotLegends→{"Male
in 1st class", "Male in 2nd class ", "Male in 3rd class"},FrameLabel→
{Style["Age in years",Bold,15], Style["Probability",Bold,15]}, Frame
→#6,FrameTicks→#7,GridLines→ {{20,40,60,80}},ImageSize→#8],Plot[{
CF[{#1,age,#5},"Probability"→ #6 ],CF[{#2,age,#5},"Probability"→ #6
],CF[{#3,age,#5},"Probability"→ #6 ]}, {age,0,90},PlotLegends→{"Female
in 1st class", "Female in 2nd class ", "Female in 3rd class"},FrameLabel→
{Style["Age in years",Bold,15], Style["Probability",Bold,15]}, Frame→#6,
FrameTicks→#7,GridLines→ {{20,40,60,80}},ImageSize→#8]}&["1st","2nd",
"3rd","male","female",True,All,250];
```

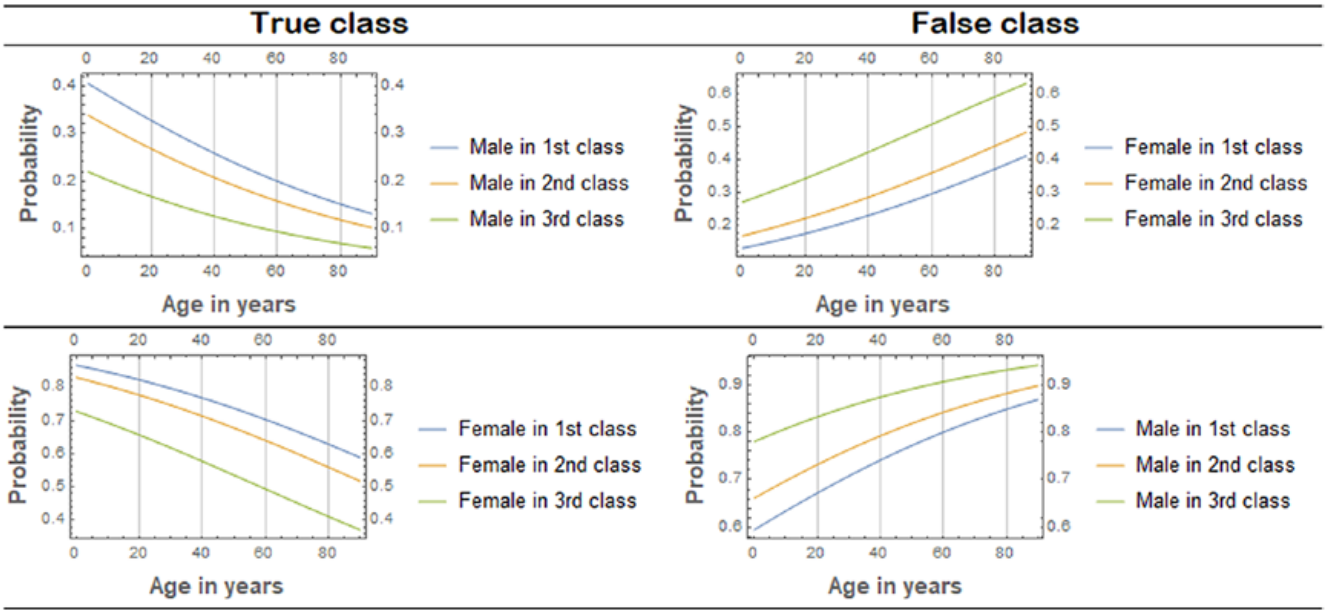


Рис. 3.40 Імовірності кожного класу в залежності від класу, віку та статі

На графіках, показаних Рис. 3.40, ясно видно, що ймовірність виживання чоловіків знижується з віком, навіть досягаючи значень нижче 20% ймовірності, чи то 1-й, 2-й чи 3-й клас. Це суперечить ймовірності виживання жінок, де вона починається зі значень вище 60% ймовірності та знижується зі збільшенням віку, досягаючи значень вище 50% для 1-го класу.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

При написанні кваліфікаційної роботи та розробці програмного виконуваного коду у Wolfram Mathematica використовувався персональний комп’ютер, тому слід зазначити, що при роботі за комп’ютерною технікою необхідно дотримуватися вимог охорони праці з ціллю збереження здоров'я.

При роботі з комп’ютером значним чином відбувається вплив на нервово-емоційний стан операторів, така робота характеризується великим навантаженням на м’язи рук при роботі з клавіатурою комп'ютера, високою інтенсивністю зорової роботи та значним розумовим перенапруженням.

Отже, раціональне планування робочого місця повинно забезпечити: зменшення втоми працівників та підвищення продуктивності праці, уникнення загального дискомфорту, якнайкраще розташування інструментів та предметів праці.

Для того, щоб виявити та проаналізувати шкідливі і небезпечні виробничі фактори необхідно почати з аналізу дотримання вимог, встановлених санітарними правилами і нормами [ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»] для виробничих приміщень та робочих місць.

На виробництві для дійсної оцінки умов праці відбувається сертифікація робочих місць відповідно до умов праці та використовується «Гігієнічна класифікація праці» за показниками небезпечності та шкідливості факторів виробничого середовища, напруженості та тяжкості трудового процесу. На основі принципів гігієнічної класифікації умови праці поділяють на чотири класи:

- Перший клас - оптимальні умови праці - це умови, за яких створюються передумови для підтримки високого рівня працездатності;

- Другий клас - допустимі умови праці, що характеризуються такими рівнями факторів виробничого середовища та трудового процесу, які не перевищують встановлених норм;

- Третій клас - шкідливі умови праці, що характеризуються такими рівнями шкідливих виробничих факторів, які перевищують нормативи і можуть мати несприятливий вплив на організм працівника і /або його нащадків;

- Четвертий клас – небезпечні/екстремальні умови праці.

На підставі сертифікації робочого місця необхідно охарактеризувати інтенсивність робіт за такими напрямами:

- відповідність обладнання нормативно-технічним вимогам;
- документації, а також характер і обсяг виконуваних робіт;
- відповідність площі та обсягу займаного робочого місця чинним нормам;
- спеціалізоване устаткування робочого місця (засоби захисту пристроїв та їх технічний стан);
- відповідність технологічного процесу, інструментів, устаткування, засобів контролю вимогам стандартів безпеки і нормам охорони праці.

Обладнання та організація робочого місця ВДТ ЕОМ повинні задовольняти відповідність конструкції всіх елементів робочого місця та їх відносного розташування ергономічним вимогам з урахуванням особливостей та характеру трудової діяльності [ДСТУ 7299:2013 «Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки»].

Раціональне планування робочого місця повинно забезпечити: найкраще розміщення інструментів та предметів праці, уникати загального дискомфорту, зменшити втому працівників та підвищити продуктивність праці.

Заходи щодо усунення ризику ураження електричним струмом зводяться до правильного розміщення обладнання та електричних кабелів. Також інші заходи для забезпечення електробезпеки збігаються із загальними заходами пожежної та електробезпеки.

В якості заходів профілактики для того, щоб забезпечити пожежну безпеку необхідно у приміщеннях використовувати приховану електромережу, ввімкнення та вимкнення живлення виконувати обладнанням за допомогою стандартних вимикачів, надійні розетки з пожежобезпечних матеріалів. Необхідно регулярно чистити внутрішні частини комп'ютерів та інше обладнання від пилу, комп'ютери розміщувати на окремих неспалюваних столах. Для запобігання іскроутворення необхідно рідше вставляти і виймати вилки з розеток. Щодо розташування робочих місць, то вони мають бути розміщені на відстані, що не є меншою 1,5 м від стіни з вікнами, та від інших стін на відстані близько 1 м, між бічними поверхнями ВДТ - 1,2 м; від задньої площини одного ВДТ до іншого екрану - 2,5 м.

Конструкція робочої поверхні користувача ВДТ повинна забезпечувати підтримку оптимального робочого положення. Сприятливе робоче положення в процесі роботи за комп’ютером забезпечується налаштуванням висоти робочого столу, підставки для ніг і стільця.

Важливою є форма спинки стільця, яка повинна повторювати форму спини працівника. Висота стільця має бути такою, щоб користувач не відчував тиску на стегна чи куприк. Бажано обладнати крісло підлокітниками. Потрібно встановити їх так, щоб не довелося тягнутися до клавіатури.

Порівняно з вікнами робоче місце доцільно розмістити таким чином, щоб на нього природне світло потрапляло з бокової частини, переважно зліва. Робочі зони потрібно розташовувати так, щоб пряме світло не потрапляло в очі. Рекомендовано розміщувати джерела світла по обидва боки екрану, паралельно напрямку погляду. Для того, щоб уникнути відблисків на екрані, клавіатурі в напрямку очей користувача, від загальних освітлювальних приладів або сонячного світла, потрібно використовувати спеціалізовані фільтри для екранів, захисні навіси, антипроблискові сітки та жалюзі на вікнах.

Екран ВДТ повинен бути розташовуваний на оптимальній відстані від очей користувача, яка становить від 600 до 700 мм, але не ближче 600 мм, враховуючи розмір буквено-цифрових символів та знаків.

При обладнанні робочого місця ВДТ лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам [ДСанПІН 3.3.2.007-98].

Дотримання всіх необхідних заходів з охорони праці забезпечує комфортні умови праці та відсутність шкоди для здоров’я, що сприяє підвищенню продуктивності праці, а також меншому виснаженню при роботі за персональним комп’ютером.

4.2 Безпека в надзвичайних ситуаціях

У процесі написання кваліфікаційної роботи та розробки програмного забезпечення в Wolfram Mathematica при роботі за персональним комп’ютером існують такі види небезпеки:

- порушення опорно-рухового апарату через тривалі не динамічні навантаження при роботі з ПК;
- незадовільні ергономічні характеристики робочого місця внаслідок нераціонального планування робочого місця, може призвести до уражень електричним струмом, механічних травм, та розладів опорно-рухового апарату;
- ризик ураження електричним струмом через недотримання правил електробезпеки або несправність електроприладів;
- негативний вплив недостатнього освітлення робочої зони на зір і продуктивність праці користувача ЕОМ, внаслідок несправності освітлювальних приладів або неправильної конструкції системи освітлення;
- негативний вплив незадовільних параметрів повітряного середовища робочої зони на здоров’я працівника, в результаті неправильної конструкції вентиляційної системи або її несправності;
- нервово-психічні перевантаження через постійні контакти з колегами по роботі, клієнтами, керівництвом при вирішенні робочих питань, які можуть

носити конфліктний характер і призвести до емоційної нестабільності та дискомфорту, внутрішнього подразнення, та захворювань нервової системи;

- негативний вплив підвищеного рівня шуму на психо-емоційний стан працівника, що пов’язано із використанням застарілого периферійного обладнання, освітлювальних приладів, копіювального обладнання, кондиціонерів;
- ризик пожежі внаслідок несправності електрообладнання, недотримання або порушення правил пожежної безпеки обслуговуючим персоналом, що може призвести до пожежі та неправильні дії персоналу в надзвичайних ситуаціях.

Відповідно до «Правил улаштування електроустановок» реалізовані такі групи заходів електробезпеки: конструктивні заходи забезпечують захист від випадкового контакту з струмопровідними частинами за допомогою їх ізоляції та захисних оболонок. Оскільки згідно з [НПАОП 40.1-1.32-01 "Правила встановлення електроустановок. Електричне обладнання спеціальних установок"] офісні приміщення в основному відносяться до класу пожежонебезпечної зони П-IIа (приміщення, в яких розташовані тверді горючі речовини), тому ступінь забезпечується захист ізоляції обладнання IP44.

Приміщення, в яких працюють розробники програмного забезпечення, класифікуються як приміщення без підвищеного ризику ураження електричним струмом. Обладнання, яке використовується в цих кімнатах, є споживачем електроенергії, що живиться від мережі змінного струму 220 В від мережі із заземленою нейтраллю, і належить до електричних установок до 1000 В закритої конструкції.

Іншим важливим завданням безпеки в надзвичайних ситуаціях є створення заходів пожежної безпеки. Закон України «Про пожежну безпеку» визначає загальні правові, соціальні і економічні основи забезпечення пожежної безпеки на території України, регулює відносини між державними органами, юридичними та фізичними особами у цій галузі, незалежно від виду їх діяльності та форм права власності.

Пожежна безпека - це стан об’єкта, при якому можливість виникнення та розвитку пожежі та вплив її небезпечних факторів на людей виключається з регульованою ймовірністю, а також забезпечується захист матеріальних цінностей. Для забезпечення пожежної безпеки в закладах проводиться протипожежна профілактика, що включає комплекс організаційно-технічних заходів, спрямованих на забезпечення безпеки людей, запобігання пожежі, обмеження її поширення, а також створення умов для успішного подолання пожежі.

Для ліквідації пожежі на початковій стадії її розвитку персонал об’єктів повинен використовувати первинне обладнання для пожежогасіння. До них відносяться:

- вогнегасники;
- протипожежне обладнання (покривала з негорючих теплоізоляційних тканин, пожежні відра, ящики з піском, лопати, ломи, сокири і тп);
- автоматичні системи пожежогасіння.

Первинне обладнання пожежогасіння, може розташовуватися як окремо, так і у складі протипожежних щитів. Це залежить від агрегатного стану та особливостей горіння різних горючих речовин та матеріалів пожежі [ДБН В. 1.1.7-2016 «Пожежна безпека об’єктів будівництва. Загальні вимоги»] поділяються на відповідні класи. Офісні приміщення містять дерев'яні меблі, електронне обладнання і паперові матеріали. Клас пожежі в офісному приміщенні [ДБН В.1.1-7:2016 «Пожежна безпека об`єктів будівництва»] - пожежі твердих речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, пластик, папір) і визначається як клас А.

Категорія приміщень [ДСТУ Б В.1.1-36:2016 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою»] - визначається як категорія D. Визначення типу та розрахунок кількості первинних засобів пожежогасіння обладнання [ДБН В.1.1-7:2016 «Пожежна безпека об`єктів будівництва. Загальні вимоги»]. Крім того, адміністративні приміщення повинні бути оснащені протипожежними датчиками,

які реагують на підвищення температури, на дим та полум'я. До прикладу, такі моделі датчиків, як ДТЛ, ІТМ.

Для запобігання пожежі надзвичайно важливо правильно оцінити пожежонебезпеку будівлі, виявити небезпеку та обґрунтувати методи та засоби протипожежної охорони та захисту.

Однією з умов забезпечення пожежної безпеки є усунення можливих джерел займання.

Несправності електропроводки, електричних розеток та вимикачів можуть стати причиною запалення у лабораторії. Необхідно регулярно здійснювати плановий огляд – виявляти та усувати існуючі несправності, щоб уникнути виникнення пожежі, з цих причин.

Іншою причиною запалення у лабораторних приміщеннях може бути несправність електроприладів. Щоб усунути виникнення пожежі, із цих причин, необхідно своєчасно проводити ремонт електроприладів та не використовувати несправні електроприлади.

При обігріванні приміщень електронагрівальними приладами з відкритими нагрівальними елементами може призвести до запалення у лабораторії. З огляду на те, що в приміщенні є папір, який є легкозаймистим матеріалом - відкриті нагрівальні поверхні можуть спричинити пожежу. В лабораторії рекомендується не використовувати відкриті нагрівальні прилади задля запобігання пожежі.

Коротке замикання в електропроводці може призвести до пожежі. Електропроводка повинна бути прихованою, щоб зменшити ймовірність пожежі внаслідок короткого замикання.

Ще однією з причин запалення є влучення блискавки у будівлю. Влітку під час грози може потрапити блискавка в будинок, що призведе до можливої пожежі. Щоб уникнути цього, рекомендується встановити громовідвід на даху будинку.

Недотримання заходів пожежної безпеки та куріння в приміщенні також можуть спричинити пожежу. Для ліквідації пожежі в результаті паління в лабораторії необхідно категорично заборонити куріння і дозволяти це лише в суворо відведеному місці.

Перед початком роботи, з метою запобігання пожежі, необхідно провести інструктаж з протипожежних заходів з працівниками, що працюють у лабораторії, де необхідно ознайомити працівників з правилами пожежної безпеки, а також навчити їх користуватися первинним обладнанням для гасіння пожежі.

Перше, що необхідно зробити під час виникнення пожежі це відключити електропостачання, зателефонувати до пожежної охорони, евакуювати людей з приміщень відповідно до плану евакуації, та розпочати гасіння полум'я вогнегасниками. Якщо є незначне джерело полум'я, тоді потрібно, використавши підручні засоби, зупинити доступ повітря до джерела займання.

Під час роботи комп'ютерів забороняється ремонтувати їх на робочому місці, не допускається працювати на пошкодженому обладнанні або захаращувати робочі місця матеріалами, які не використовуються для поточних робіт.

Ремонт, будь-яке технічне обслуговування та налагодження комп'ютера, а також інші операції з цього приводу слід проводити лише тоді, коли живлення повністю вимкнено. У тих випадках, коли ремонт та інші процедури неможливо виконати при відключенні електроживлення, необхідно, щоб обладнання та периферійні пристрої були заземлені, ремонт виконували двоє або більше робітників з використанням інструментів з ізольованими ручками, а також килимки-діелектрики на підлогу.

Режим праці та відпочинку працівників електронно-обчислювальної техніки визначається [ДСанНіП 3.3.2-007-98]. Під час робочого дня кожні 40-50 хвилин роботи потрібно робити перерви для відпочинку, які тривають близько 3-5-хвилин. Загальна тривалість роботи на день не повинна перевищувати 4 години, а на тиждень - 20 годин.

Таким чином, при дотриманні правил безпеки в надзвичайних ситуаціях, можна запобігти виникненню критичних результатів.

Програмний продукт з використанням Wolfram Mathematica, що описаний у кваліфікаційній роботі розроблений з урахуванням вимог техніки безпеки та пожежної безпеки.

ВИСНОВКИ

В атестаційні роботі розвинено концепцію застосування методів машинного навчання та засобів обробки статистичної інформації з використанням мов програмування Wolfram Language та середовища Wolfram Mathematica. Встановлено, що методи роботи із статистичними даними можуть бути ефективно реалізованими у середовищі Wolfram Mathematica та можуть ефективно застосовуватись до масивів даних різного походження з довільними обсягами інформації.

Методи математичного навчання застосовано до набору даних з відомими вхідними параметрами. Виконано аналіз інформацію тестового набору даних та встановлено наявність кореляції між його параметрами. Показано, що середовище Wolfram Mathematica є зручним середовищем для реалізації машинного навчання та роботи з даними й може бути інтегрованим в широкий спектр складних програмних систем.

Отримані результати мають практичне значення та будуть корисними програмістам працюючим у галузі інженерії даних, науки про дані, машинного навчання, а також можуть бути застосованими у різноманітних галузях таких як нейронаука, медична інформатика, економічна статистика з метою оптимізації праці та прогнозування результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO Central Secretary. Information technology – Vocabulary. Standard ISO/IEC 2382:2015. Genf, Schweiz: International Organization for Standardization, 2015, p. 2121272.

2. Merriam–Webster. Data. Accessed Apr. 10, 2022. url: https://www.merriam-webster.com/dictionary/data.

3. Clifford M. Will. Theory and Experiment in Gravitational Physics. Cambridge University Press, Sept. 2018. doi: 10.1017/9781316338612.

4. Holger Ernst. “Patent information for strategic technology management”. In: World Patent Information 25.3 (Sept. 2003), pp. 233–242. doi: 10. 1016/s0172-2190(03)00077-2.

5. Peter Walde et al. “Erstellung von Technologie- und Wettbewerbsanalysen mithilfe von Big Data”. In: Wirtschaftsinformatik & Management 5.2 (Feb. 2013), pp. 12–23. doi: 10.1365/s35764-013-0274-7.

6. Robert Follmer and Dana Gruschwitz. Mobility in Germany – short report (edition 4.0). Accessed Apr. 18, 2022. Bonn, Berlin, Sept. 2019. url: http: //www.mobilitaet-in-deutschland.de/pdf/MiD2017_ShortReport.pdf.

7. Fábio Duarte and Ricardo Álvarez. “The data politics of the urban age”. In: Palgrave Communications 5.1 (May 2019). doi: 10.1057/s41599-019- 0264-3.

8. Deutscher Wetterdienst – Zentraler Vertrieb Klima und Umwelt. Klimadaten Deutschland. Accessed Apr. 1, 2020. Offenbach. url: https://www. dwd.de/DE/leistungen/klimadatendeutschland/klimadatendeutschland. html.

9. Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. 3rd ed. Elsevier, 2012.

10. Merriam-Webster. Data mining. Accessed Apr. 10, 2022. url: https: //www.merriam-webster.com/dictionary/data%20mining.

11. R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. 2020. url: https://www.Rproject.org/.

12. Guido van Rossum and Fred L. Drake. Python 3 Reference Manual. Scotts Valley, USA: CreateSpace, 2009.

13. Hadley Wickham. ggplot2. Elegant Graphics for Data Analysis. Springer, New York, 2009. doi: 10.1007/978-0-387-98141-3.

14. Till Tantau. The TikZ and PGF Packages. Manual for version 3.1.7. Nov. 2020. url: https://pgf-tikz.github.io/pgf/pgfmanual.pdf.

15. Simon Urbanek and Jeffrey Horner. Cairo: R Graphics Device using Cairo Graphics Library for Creating High-Quality Bitmap (PNG, JPEG, TIFF), Vector (PDF, SVG, PostScript) and Display (X11 and Win32) Output. R package. 2020. url: https://CRAN.R-project.org/package=Cairo.

16. Claus O. Wilke. cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'. R package. 2020. url: https : / /CRAN.R- project . org / package=cowplot.

17. Hadley Wickham et al. dplyr: A Grammar of Data Manipulation. R package. 2020. url: https://CRAN.R-project.org/package=dplyr.

18. Winston Chang. extrafont: Tools for using fonts. R package. 2014. url: https://CRAN.R-project.org/package=extrafont.

19. Daniel Müllner. "fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python". In: Journal of Statistical Software 53.9 (2013), pp. 1–18. url: http://www.jstatsoft.org/v53/i09/.

20. Alina Beygelzimer et al. FNN: Fast Nearest Neighbor Search Algorithms and Applications. R package. 2019. url: https://CRAN.R-project.org/ package=FNN.

21. Hadley Wickham. forcats: Tools for Working with Categorical Variables (Factors). R package. 2020. url: https://CRAN.R-project.org/package= forcats.

22. Andrie de Vries and Brian D. Ripley. ggdendro: Create Dendrograms and Tree Diagrams Using 'ggplot2'. R package. 2020. url: https://CRAN.Rproject. org/package=ggdendro.

23. Thomas Lin Pedersen. ggforce: Accelerating 'ggplot2'. R package. 2020. url: https://CRAN.R-project.org/package=ggforce.

24. Kamil Slowikowski. ggrepel: Automatically Position Non-Overlapping Text Labels with 'ggplot2'. R package. 2020. url: https://CRAN.R-project.org/ package=ggrepel.

25. Herve Cardot. Gmedian: Geometric Median, k-Median Clustering and Robust Median PCA. R package. 2020. url: https://CRAN.R-project. org/package=Gmedian.

26. Baptiste Auguie. gridExtra: Miscellaneous Functions for 'Grid' Graphics. R package. 2017. url: https://CRAN.R-project.org/package=gridExtra.

27. Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". In: InterJournal Complex Systems (2006), p. 1695. url: https://igraph.org.

28. Stefano Meschiari. latex2exp: Use LaTeX Expressions in Plots. R package. 2015. url: https://CRAN.R-project.org/package=latex2exp.

29. Garrett Grolemund and Hadley Wickham. "Dates and Times Made Easy with lubridate". In: Journal of Statistical Software 40.3 (2011), pp. 1–25. url: https://www.jstatsoft.org/v40/i03/.

30. Jeroen Ooms. magick: Advanced Graphics and Image-Processing in R. R package. 2020. url: https://CRAN.R-project.org/package=magick.

31. Doug McIlroy et al. mapproj: Map Projections. R package. 2020. url: https://CRAN.R-project.org/package=mapproj.

32. W. N. Venables and B. D. Ripley. Modern Applied Statistics with S. 4th ed. Springer, New York, 2002. doi: 10.1007/978-0-387-21706-2.

33. Friedrich Leisch and Evgenia Dimitriadou. mlbench: Machine Learning Benchmark Problems. R package. 2010. url: https://CRAN.R-project. org/package=mlbench.

34. Alan Genz et al. mvtnorm: Multivariate Normal and t Distributions. R package. 2020. url: https://CRAN.R-project.org/package=mvtnorm.

.35. Alan Genz and Frank Bretz. Computation of Multivariate Normal and t Probabilities. Lecture Notes in Statistics. Springer, Berlin, Heidelberg, 2009.

36. Stefan Fritsch, Frauke Günther, and Marvin N.Wright. neuralnet: Training of Neural Networks. R package. 2019. url: https://CRAN.R-project.org/ package=neuralnet.

37. Diego Hernangómez. nominatimlite: Interface with ’Nominatim’ API Service. R package. 2022. doi: 10.5281/zenodo.5113195. url: https: //dieghernan.github.io/nominatimlite/.

38. David Meyer and Christian Buchta. proxy: Distance and Similarity Measures. R package. 2020. url: https://CRAN.R-project.org/package= proxy.

39. Damian W. Betebenner. randomNames: Function for Generating Random Names and a Dataset. R package. 2019. url: https://cran.r-project.org/ package=randomNames.

40. Hadley Wickham. “Reshaping Data with the reshape Package”. In: Journal of Statistical Software 21.12 (2007), pp. 1–20. url: http://www.jstatsoft. org/v21/i12/.

41. Michael W. Kearney. “rtweet: Collecting and analyzing Twitter data”. In: Journal of Open Source Software 4.42 (2019). R package, p. 1829. doi: 10.21105/joss.01829.

42. Hadley Wickham and Dana Seidel. scales: Scale Functions for Visualization. R package. 2020. url: https://CRAN.R-project.org/package=scales.

43. Carter T. Butts. sna: Tools for Social Network Analysis. R package. 2020. url: https://CRAN.R-project.org/package=sna.

44. Edzer J. Pebesma and Roger S. Bivand. “Classes and methods for spatial data in R”. In: R News 5.2 (Nov. 2005), pp. 9–13. url: https://CRAN.Rproject. org/doc/Rnews/.

45. Roger S. Bivand, Edzer Pebesma, and Virgilio Gomez-Rubio. Applied spatial data analysis with R. 2nd ed. Springer, New York, 2013. url: https://asdar-book.org/.

46. Mark P. J. van der Loo. “The stringdist package for approximate string matching”. In: The R Journal 6 (1 2014), pp. 111–122. url: https:// CRAN.R-project.org/package=stringdist.

47. Hadley Wickham. stringr: Simple, Consistent Wrappers for Common String Operations. R package. 2019. url: https://CRAN.R-project.org/ package=stringr.

48. Hadley Wickham et al. “Welcome to the tidyverse”. In: Journal of Open Source Software 4.43 (2019), p. 1686. doi: 10.21105/joss.01686.

49. Julia Silge and David Robinson. “tidytext: Text Mining and Analysis Using Tidy Data Principles in R”. In: JOSS 1.3 (2016). doi: 10.21105/joss.00037.

50. Justin Donaldson. tsne: t-Distributed Stochastic Neighbor Embedding for R (t-SNE). R package. 2016. url: https://CRAN.R-project.org/package= tsne.

51. Kyle Bittinger. usedist: Distance Matrix Utilities. R package. 2020. url: https://CRAN.R-project.org/package=usedist.

ДОДАТКИ

ДОДАТОК А

Публікація в науковому виданні