Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» грудня 2024 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**МОДЕЛЬ КОМП'ЮТЕРНОЇ СИСТЕМИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ЗЛОЧИННОСТІ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ**»

Спеціальність 123 – Комп'ютерна інженерія.
Галузь знань: 12 – Інформаційні технології.
Освітня програма «Комп'ютерна інженерія».

Захищено на засіданні
Екзаменаційної комісії № 42
протокол № __ від __.12.2024 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
Студент групи КІ – 61
Румянцев Данило Максимович____

Керівник: к.т.н, доцент, доцент кафедри комп'ютерних систем та робототехніки
Бакуменко Ніна Станіславівна _____

Рецензент: д.т.н, професор, професор кафедри комп'ютерної математики і аналізу даних Національного технічного університету «Харківський політехнічний інститут»
Погорєлов Станіслав Вікторович ____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 71 сторінка, із яких 51 сторінка основної частини з 31 рисунком, 2 таблицями, списку використаних джерел із 21 найменувань та чотирма додатками.

Метою кваліфікаційної роботи є порівняння ефективності використання методів машинного навчання, таких як лінійна регресія, дерева рішень, алгоритм k-найближчих сусідів та нейронні мережі для аналізу та прогнозування злочинності.

Об'єкт дослідження – процес встановлення тенденцій та закономірностей розвитку злочинності за допомогою аналізу соціо-економічних та демографічних показників.

Предмет дослідження – методи машинного навчання для ідентифікації залежностей між змінними, зокрема, регресійні, нейромережеві та ін.

Актуальність роботи полягає в необхідності здійснення довгострокового та оперативного аналізу статистичної інформації з подальшим прогнозуванням факторів та чинників, які впливають на показники злочинності. Сучасні методи регресійного аналізу надають інструменти для вивчення залежностей між демографічними статистичними показниками і скоєними злочинами. Отримані результати можуть допомогти у вивченні проблеми аналізу впливу на злочинність соціальних, демографічних чинників, що дозволить планувати профілактичні заходи, розподіляти ресурси правоохоронних органів більш ефективно та ін.

Область застосування – розроблений програмний продукт можна використовувати для подальшого дослідження аналізу та прогнозування злочинності.

Ключові слова: методи машинного навчання, прогнозування злочинності, лінійна регресія, регресія Ласо, гребнева регресія, дерева рішень, метод k-найближчих сусідів, нейронні мережі.

ABSTRACT

The explanatory note to the master's attestation work consists of an introduction, three sections, conclusions, a list of sources used and four appendices. The total volume of the work is 71 pages, of which 51 pages of the main part with 31 figures, 2 tables, a list of sources used with 21 names and four appendices.

The purpose of the qualification work is to compare the effectiveness of using machine learning methods, such as the linear regression, decision trees, k-nearest neighbors algorithm and neural networks for analyzing and predicting crime.

The object of the study is the process of establishing trends and patterns of crime development through the analysis of socio-economic and demographic indicators.

The subject of the study is machine learning methods for identifying dependencies between variables, in particular, regression, neural network, etc.

The relevance of the work lies in the need to carry out long-term and operational analysis of statistical information with subsequent forecasting of factors and factors that affect crime rates. Modern methods of regression analysis provide tools for studying the relationships between demographic statistical indicators and committed crimes. The results obtained can help in studying the problem of analyzing the impact of social and demographic factors on crime, which will allow planning preventive measures, distributing law enforcement resources more effectively, etc.

Keywords: machine learning methods, crime prediction, linear regression, Lasso regression, ridge regression, decision trees, k-nearest neighbors method, neural networks.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

НМ - нейронна мережа;

РБНМ - радіально-базисна нейронна мережа;

РБФ - радіально-базисна функція;

КНС - K-найближчих сусідів.

ВСТУП

З розвитком сучасних технологій та зі зростанням інформації у навколишньому світі, яку можна використовувати або знаходити, виникає проблема з її обробкою «вручну». Тому для цього використовують моделі побудовані за методами машинного навчання, і правильне навчання цих моделей стає актуальним завданням, особливо при роботі з конфіденційними даними.

Актуальність роботи. У наш час, злочинність документується у вигляді таблиць даних, і зі зростанням потужності сучасних технологій, виникає можливість аналізу, дослідження і прогнозування злочинності за допомогою таких аналітичних інструментів як машинне навчання і глибоке навчання. Завдяки алгоритмам машинного навчання, таким як дерева рішень або випадкові ліси, можна знаходити тенденції злочинності, а також кореляції між злочинами та екологічними або демографічними факторами.

Необхідність здійснення довгострокового та оперативного аналізу статистичної інформації з подальшим прогнозуванням факторів та чинників, які впливають на показники злочинності, обумовлює актуальність даної роботи. Сучасні методи регресійного аналізу надають інструменти для вивчення залежностей між демографічними статистичними показниками і скоєними злочинами. Отримані результати можуть допомогти у вивченні проблеми аналізу впливу на злочинність соціальних, демографічних чинників, що дозволить планувати профілактичні заходи, розподіляти ресурси правоохоронних органів більш ефективно та ін.

Звичайно, ця проблема не є простою, оскільки кореляція не є доказом зв’язку, і потребує коректної інтерпретації експертом, а при неправильних висновках існують дуже значні ризики. Ще однією проблемою є отримання високоякісного набору даних про злочинність, які, зазвичай, дуже важко отримати, а зі збором і використанням цих даних існує багато етичних і

конфіденційних питань. Але завдяки сучасним досягненням у галузі машинного навчання є можливість розв’язати дану проблему.

Використання методів машинного навчання для аналізу та прогнозування злочинності не є новиною. Так у роботі [1] були розроблені моделі методом k-найближчих сусідів та деревом рішень, посилаючись на дані злочинів у Ванкувері. Ці моделі мали точність від 39% до 44%, що може здаватися досить низкою, але ці моделі є гарною основою для подальших досліджень. Якщо розглянути роботу [2], то можна побачити детальний опис проблем, які виникають при прогнозуванні злочинів. У цій роботі автори проаналізували понад 150 статей з різними алгоритмами прогнозування на різних наборах даних, а також привернули увагу до подальших досліджень у цій сфері. Незважаючи на всі розглянуті алгоритми машинного навчання та глибокого навчання, ця робота не розглядала регресійний аналіз і нейронні мережі, останні з яких теоретично можуть показувати кращі результати.

Прикладом використання нейронних мереж для прогнозування злочинності може стати стаття [3], у якій одна з моделей нейронних мереж визначала місце злочину в межах сусіднього поштового індексу у 31,2% випадків, коли місце злочину було одразу невідомо. Наприкінці статті автор спонукає до подальшого вивчення використання нейронних мереж для прогнозу злочинності, особливо при використанні певних додаткових ознак для покращення прогнозів.

Метою кваліфікаційної роботи є порівняння ефективності використання методів машинного навчання, таких як лінійна регресія, дерева рішень, алгоритм k-найближчих сусідів та нейронні мережі для аналізу та прогнозування злочинності.

Об’єкт дослідження – процес встановлення тенденцій та закономірностей розвитку злочинності за допомогою аналізу соціо-економічних та демографічних показників.

Предмет дослідження – методи машинного навчання для ідентифікації залежностей між змінними, зокрема, регресійні, нейромережеві та ін.

Завдання дослідження

1. Сформулювати задачу аналізу та прогнозування даних
2. Здійснити порівняльний аналіз методів аналізу та прогнозування даних
3. Розробити математичні моделі аналізу та прогнозування злочинності за допомогою методів машинного навчання.
4. Розробити програмно-алгоритмічні реалізації моделей аналізу та прогнозування злочинності за допомогою методів машинного навчання.
5. Виконати тестування моделей, аналіз та інтерпретацію отриманих результатів

РОЗДІЛ 1.
ПОСТАНОВКА ЗАДАЧІ РЕГРЕСІЙНОГО АНАЛІЗУ ТА ОГЛЯД РЕГРЕСІЙНИХ МЕТОДІВ

Регресійний аналіз є фундаментальною концепцією в галузі машинного навчання. Це підпадає під контрольоване навчання, у якому алгоритм навчається як з вхідними функціями, так і з вихідними мітками. Це допомагає встановити зв’язок між змінними, оцінюючи, як одна змінна впливає на іншу.

Регресія в машинному навчанні складається з математичних методів, які дозволяють дослідникам даних передбачити безперервний результат (y) на основі значення однієї або кількох змінних предиктора (x). Прикладом регресії у машинному навчанні може бути лінійна регресія – одна з найпопулярніших форм регресійного аналізу через її легкість використання для передбачення та прогнозування.

1.1 Оцінка регресійного алгоритму машинного навчання

Нехай ми маємо модель, яка передбачає температуру наступного тижня, де прогнозована температура залежить від різних властивостей, таких як вологість, атмосферний тиск, температура повітря та швидкість вітру. Для оцінки точності прогнозів за цією регресійною моделлю враховують два важливі показники: дисперсію та зміщення.

Дисперсія – це величина, на яку змінюється оцінка цільової функції, якщо використовуються інші навчальні дані. Цільова функція f встановлює зв'язок між вхідними і вихідними змінними. Коли використовується інший набір даних, цільова функція має залишатися стабільною з невеликою дисперсією, оскільки для будь-якого заданого типу даних модель має бути загальною. У цьому випадку прогнозована температура змінюється на основі варіацій у навчальному наборі даних. Щоб уникнути хибних прогнозів, нам

потрібно переконатися, що дисперсія низька. З цієї причини модель має бути узагальнена, щоб прийняти невидимі особливості температурних даних і створити кращі прогнози.

Зміщення — це схильність алгоритму постійно вивчати неправильні речі, не беручи до уваги всю інформацію в даних. Щоб модель була точною, зміщення має бути низьким. Якщо в наборі даних є невідповідності, тобто певні відсутні значення або помилки у вхідних даних, зміщення буде високим, а прогнозована температура буде неправильною.

Зміщення і дисперсія завжди знаходяться в компромісі. Коли зміщення велике, дисперсія низька, а коли дисперсія низька, зміщення велике. Перший випадок виникає, коли модель занадто проста з меншою кількістю параметрів, а другий – коли модель складна з великою кількістю параметрів. Ми вимагаємо, щоб як дисперсія, так і зміщення були якомога меншими, і щоб досягти цього, компроміс потрібно розглядати обережно, тоді він підніметься до бажаної кривої.

Точність і похибка є ще двома важливими показниками. Похибка – це різниця між фактичним значенням і прогнозованим значенням, оціненим моделлю. Точність – це частка прогнозів, які модель отримала правильно.

Щоб модель була ідеальною, очікується, що вона матиме низьку дисперсію, низьке зміщення та низьку похибку. Щоб досягти цього, потрібно розділити набір даних на навчальні та тестові набори даних. Потім модель вивчатиме шаблони з навчального набору даних, а продуктивність буде оцінюватися на тестовому наборі даних.

Щоб зменшити похибку під час навчання моделі, існують функції помилки. Якщо модель запам’ятовує передані їй навчальні дані, але не знаходить закономірності, вона видаватиме хибні прогнози щодо невидимих даних. Тоді крива, отримана з навченої моделі, проходитиме через усі точки

даних, а точність тестового набору даних буде низькою. Це називається перенавчанням і спричинено високою дисперсією.

З іншого боку, якщо модель добре працює на тестових даних, але з низькою точністю на навчальних даних, це призводить до недостатнього підбору.

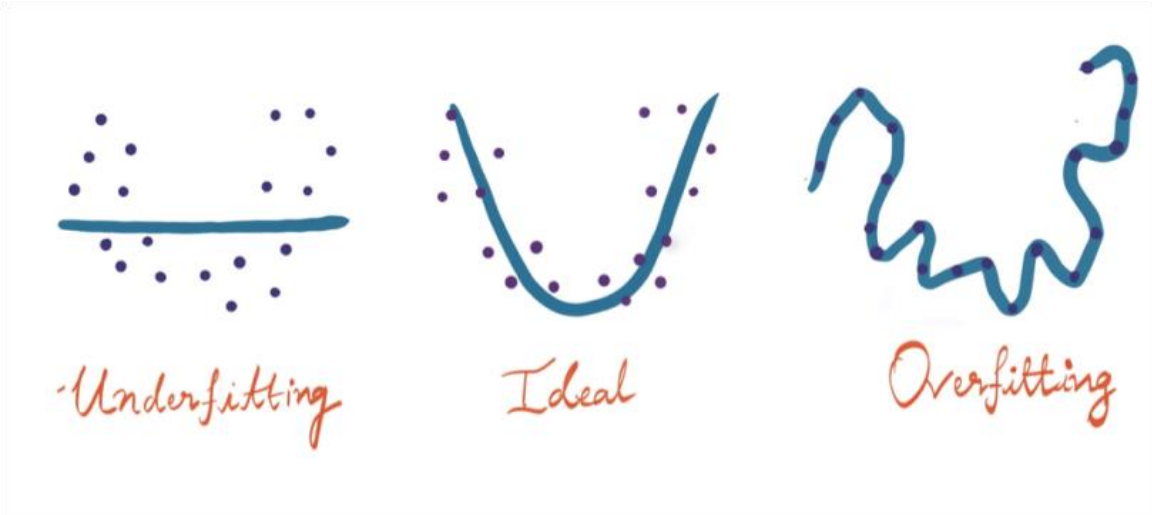


Рис. 1.1 – Зображення проблеми недостатнього навчання і перенавчання моделі

1.2 Лінійна регресія в машинному навчанні

Лінійна регресія знаходить лінійний зв’язок між залежною змінною та однією чи кількома незалежними змінними за допомогою прямої лінії, яка найкраще підходить. Як правило, лінійна модель робить прогноз шляхом простого обчислення зваженої суми вхідних характеристик плюс константу, яка називається зміщенням (також називається перехопленням). У цій техніці залежна змінна є безперервною, незалежна змінна (або змінні) може бути безперервною або дискретною, а природа лінії регресії є лінійною. Математично прогноз із використанням лінійної регресії подається як:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Тут y — прогнозоване значення,

n - загальна кількість вхідних функцій,

x_i — це вхідна функція для значення $i \in (1; n)$,

θ_i — параметр моделі (θ_0 — зміщення, а коефіцієнти — $\theta_1, \theta_2, \dots \theta_n$).

Зміщення – це відхилення, викликане рівнянням лінії y = mx для передбачень. Потрібно налаштувати зміщення так, щоб змінювати положення лінії, яке найкраще підходить для даних.

1.2.1 Проста лінійна регресія в машинному навчанні

Проста лінійна регресія є одним із найпростіших, але потужних методів регресії. Вона має одну вхідну (x) та одну вихідну змінні (y) і допомагає передбачити вихідні дані з навчених вибірок, підбираючи пряму лінію між цими змінними. Наприклад, ми можемо передбачити оцінку студента на основі кількості годин, які він/вона навчається, використовуючи просту лінійну регресію.

Математично це представлено рівнянням:

$$y = mx + c$$

де x – незалежна змінна (вхід),

y – залежна змінна (вихід),

m – нахил,

і c є параметром.

У простій лінійній регресії припускається, що нахил і відрізок є коефіцієнтом і зміщенням відповідно. Вони діють як параметри, які впливають на положення лінії, яка буде побудована між даними.

Нехай точки даних нанесено різними кольорами. Нижче наведено зображення, яке показує найкращу лінію, проведену за допомогою лінійної регресії.

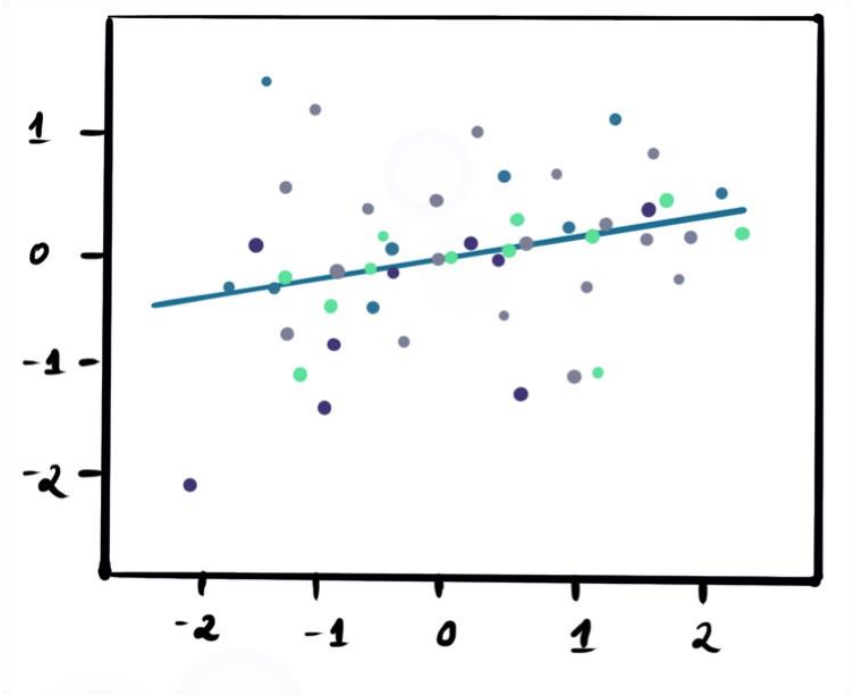


Рис. 1.2 – Приклад простої лінійної регресії

1.2.2 Множинна лінійна регресія в машинному навчанні

Множинна лінійна регресія схожа на просту лінійну регресію, але у ній існує більше однієї незалежної змінної. Кожне значення незалежної змінної x асоціюється зі значенням залежної змінної y. Оскільки це багатовимірне представлення, найкраща лінія є площиною.

Математично це виражається так:

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$$

Нехай потрібно передбачити, чи здасть студент іспит чи ні. Тут можна враховувати кілька вхідних даних, наприклад кількість годин, які він/вона витратив на навчання, загальну кількість предметів і години, які він/вона спав минулої ночі. Оскільки ми маємо кілька вхідних даних, то використовуємо множинну лінійну регресію.

1.3 Використання регресійного аналізу

Регресійний аналіз — це статистичний метод, який використовується для вивчення зв’язку між залежною змінною та однією або кількома незалежними змінними. Він використовується для різних цілей, зокрема:

- Передбачення**:** регресійний аналіз можна використовувати для передбачення значень залежної змінної на основі значень незалежних змінних. Наприклад, якщо ми хочемо передбачити продажі продукту на основі витрат на рекламу та розміру ринку, ми можемо використати регресійний аналіз, щоб визначити зв’язок між цими змінними та передбачити продажі на основі значень незалежних змінних.
- Перевірка гіпотез: регресійний аналіз можна використовувати для перевірки гіпотез про зв’язок між залежною та незалежною змінними. Наприклад, за допомогою регресійного аналізу ми можемо перевірити, чи існує значний зв’язок між курінням і раком легенів.
- Контрольні змінні: регресійний аналіз можна використовувати для контролю інших змінних, які можуть впливати на зв’язок між залежними та незалежними змінними. Наприклад, якщо ми хочемо вивчити зв’язок між доходом і здоров’ям, ми можемо захотіти контролювати такі змінні, як вік, стать і освіта.
- Прогнозування: регресійний аналіз можна використовувати для прогнозування майбутніх тенденцій на основі історичних даних. Наприклад, ми можемо використовувати регресійний аналіз, щоб спрогнозувати попит на продукт на основі минулих даних про продажі та інших відповідних змінних.

1.4 Недоліки регресійного аналізу

Регресійний аналіз, незважаючи на всі свої переваги, також має кілька проблем, з якими стикаються багато методів регресії:

- Основним обмеженням лінійних моделей є їхня гнучкість. Лінійні моделі припускають лінійний зв’язок між залежною змінною та незалежними змінними. Це припущення може бути обмеженим, особливо коли зв’язок між змінними не є лінійним. Нелінійні зв’язки часто є складнішими і можуть вимагати більш складних моделей, щоб вловити їхні нюанси.
- Моделі лінійної регресії чутливі до викидів. Викиди можуть впливати на нахил і перетин лінії регресії, що призводить до неточних прогнозів, тому важливо ідентифікувати викиди та відповідним чином обробляти їх.
- Моделі лінійної регресії припускають, що зв’язок між залежною змінною та незалежними змінними є лінійним. У деяких випадках це припущення може не відповідати дійсності, що призводить до неточних прогнозів.
- Моделі лінійної регресії схильні до перенавчання, особливо коли кількість незалежних змінних велика порівняно з розміром вибірки.
- Моделі регресії припускають, що незалежні змінні не сильно корельовані одна з одною.
- Моделі регресії не можуть безпосередньо обробляти категоріальні змінні.
- Моделі регресії припускають, що дисперсія залежної змінної постійна на всіх рівнях незалежних змінних. На практиці це припущення може не відповідати дійсності, що призводить до неточних прогнозів.

Загалом, регресійний аналіз є корисним інструментом для аналізу та розуміння зв’язку між змінними, а також для прогнозування та прийняття обґрунтованих рішень на основі цього зв’язку, але слід пам’ятати про його недоліки при побудові моделей.

1.5 Регресія Ласо

Регресія ласо, також відома як регулярізація L1, є формою регуляризації моделей лінійної регресії. Регулярізація — це статистичний метод для зменшення помилок, викликаних переобладнанням навчальних даних. Цей підхід можна відобразити такою формулою:

$$\text{w-hat} = \text{argmin}_w \, \text{MSE}(W) + ||w||_1$$

Концепції, що лежать в основі техніки Ласо, можна простежити у дослідницькій роботі з геофізики 1986 року Сантози та Саймса, у якій використано штраф L1 для коефіцієнтів. Однак у 1996 році статистик Роберт Тібширані незалежно розробив і популяризував термін, «lasso», заснований на невід’ємній роботі Бреймана.

Lasso означає найменше абсолютне скорочення та оператор вибору (Least Absolute Shrinkage and Selection Operator). Воно часто використовується в машинному навчанні для обробки даних великої розмірності, оскільки воно полегшує автоматичний вибір функцій у своїй програмі. Це робиться шляхом додавання штрафного терміну до залишкової суми квадратів (RSS), який потім множиться на параметр регуляризації (лямбда або λ). Цей параметр регулярізації контролює ступінь застосованої регулярізації. Більші значення лямбда збільшують штраф, скорочуючи більше коефіцієнтів до нуля; це згодом зменшує важливість (або взагалі виключає) деяких функцій з моделі, що призводить до автоматичного вибору функцій. І навпаки, менші значення лямбда зменшують ефект штрафу, зберігаючи більше функцій у моделі.

Це штрафування сприяє розрідженості всередині моделі, що може допомогти уникнути проблем мультиколінеарності та проблем з перенавчанням у наборах даних. Мультиколінеарність виникає, коли дві або більше незалежних змінних сильно корельовані одна з одною, що може бути проблематичним для причинно-наслідкового моделювання. Моделі надмірного пристосування будуть погано узагальнювати нові дані, що повністю зменшить їх цінність. Зменшуючи коефіцієнти регресії до нуля, ласо-регресія може ефективно усунути незалежні змінні з моделі, обходячи ці потенційні проблеми в процесі моделювання.

Регресія ласо ідеально підходить для проблем прогнозування; її здатність виконувати автоматичний вибір змінної може спростити моделі та підвищити точність прогнозування. Тим не менш, гребнева регресія може перевершити ласо-регресію через величину зміщення, яку вносить ласо-регресія шляхом зменшення коефіцієнтів до нуля. Вона також має свої обмеження щодо корельованих ознак у даних, оскільки вона довільно вибирає ознаку для включення в модель.

Регресія ласо може бути ідеальною в наступних сценаріях:

- Набір даних вважається багатовимірним, якщо кількість змінних предикторів набагато більша за кількість спостережень. Регресія ласо може допомогти зменшити розмірність у наборі даних, зменшивши вагові параметри до нуля, усуваючи менш важливі функції з моделі.
- Зміщення, внесене штрафом L1, штучно зменшить коефіцієнти до нуля. Деякі змінні скорочуються точно до нуля, залишаючи в моделі підмножину найважливіших змінних для прогнозування.

Регресія ласо може впоратися з деякою мультиколінеарністю без негативного впливу на інтерпретацію моделі, але вона не може подолати сильну мультиколінеарність. Якщо коваріати сильно корельовані, ласо-

регресія буде довільно виключати одну з ознак моделі. Еластична чиста регулярізація є хорошою альтернативою в цій ситуації.

1.6 Гребнева (ridge) регресія

Гребнева регресія — це техніка лінійної регресії, яка використовується в статистиці та машинному навчанні для вирішення проблеми мультиколінеарності, яка виникає, коли незалежні змінні в регресійній моделі сильно корельовані. Це техніка регулярізації, яка додає штрафний термін до функції вартості лінійної регресії, щоб запобігти тому, щоб модель стала занадто складною та переобладнала дані.

У традиційній лінійній регресії метою є мінімізація суми квадратів помилок між прогнозованими значеннями та фактичними цільовими значеннями. Гребнева регресія розширює це, додаючи член регуляризації до функції вартості. Функція витрат для гребневої регресії зазвичай виражається як:

*Вартість = сума квадратів помилок + λ * (сума квадратів коефіцієнтів)*

Тут λ (лямбда) є параметром регуляризації, також відомим як параметр усадки. Він контролює силу регулярізації. Коли λ дорівнює нулю, гребнева регресія еквівалентна звичайній лінійній регресії за методом найменших квадратів. Зі збільшенням λ зростає вплив члена регуляризації, що призводить до менших значень коефіцієнта, що може допомогти зменшити мультиколінеарність і надмірну підгонку.

Ключові моменти про гребневу регресію:

1 Гребнева регресія особливо корисна, коли є набір даних із висококорельованими незалежними змінними. Додаючи член

регуляризації, це зменшує вплив мультиколінеарності, роблячи модель більш стабільною та надійною.

2 Гребнева регресія вносить невелику кількість зміщення в модель в обмін на зменшення дисперсії. Цей компроміс часто призводить до кращої продуктивності узагальнення невидимих даних.

3 Гребнева регресія не виконує вибір функцій так само, як регресія Ласо, яка може встановити деякі коефіцієнти рівними точно нулю. Натомість вона зменшує коефіцієнти до нуля, роблячи всі функції потенційно релевантними.

4 Вибір параметра регуляризації λ є критичним. Перехресна перевірка зазвичай використовується для вибору відповідного значення для λ, яке врівноважує складність моделі та продуктивність даних навчання.

5 Важливо нормалізувати або стандартизувати вхідні функції перед застосуванням гребневої регресії, щоб переконатися, що термін регуляризації має послідовний вплив на всі функції.

1.7 K-найближчі сусіди (KНС)

Регресія K-найближчих сусідів (KНС) — це непараметричний алгоритм машинного навчання, який можна використовувати як для завдань класифікації, так і для регресії. Цей алгоритм робить прогнози, знаходячи K найближчих точок даних до заданого вхідного сигналу та усереднюючи їхні цільові значення (для числової регресії).

Розглянемо як працює регресія KНС:

1 Збір даних: Починаємо роботу з набору даних, який включає як вхідні функції, так і цільові значення. У завданнях регресії цільові значення є безперервними та представляють результат, який потрібно передбачити.

2 Вибір кількості сусідів (k): далі потрібно вибрати кількість найближчих сусідів k, які використовуватимуться для створення прогнозів. Це гіперпараметр, який можна налаштувати на основі характеристик даних. Маленьке k може призвести до шумних прогнозів, тоді як велике k може призвести до надмірно згладжених прогнозів.
3 Метрика відстані: КНС покладається на метрику відстані для вимірювання подібності між точками даних. Залежно від природи даних можна використовувати різні показники відстані.
4 Прогноз: якщо потрібно зробити прогноз для нової точки вхідних даних, КНС обчислює відстань між цією точкою та всіма іншими точками даних у наборі даних. Потім він вибирає k точок даних з найменшими відстанями.
5 Прогноз регресії: для регресії прогнозоване значення для нової точки даних є середнім цільових значень K-найближчих сусідів. Це може бути просте середнє арифметичне.

Регресію КНС легко реалізувати та зрозуміти, але вона може бути дорогою з точки зору обчислень, особливо для великих наборів даних, оскільки вимагає обчислення відстані між новою точкою даних і всіма існуючими точками даних. Крім того, вибір правильного значення k і відповідної метрики відстані може вплинути на якість прогнозів. Перехресна перевірка та налаштування гіперпараметрів можуть допомогти у виборі відповідних значень для k і метрики відстані.

1.8 Регресія дерева рішень (Decision Tree Regression)

Регресія дерева рішень – це метод машинного навчання, який створює деревоподібну модель для прогнозування безперервних числових значень. На

відміну від завдань класифікації, де результати є категоричними, регресія дерева рішень зосереджена на оцінці числових результатів.

Переваги регресії дерева рішень:

- Дерева рішень можуть фіксувати нелінійні зв’язки між змінними, не вимагаючи явних перетворень або припущень.
- На дерева рішень менше впливають викиди порівняно з іншими моделями регресії. Критерій розщеплення, який використовується в деревах рішень, більш стійкий до екстремальних значень.
- Дерева рішень можуть обробляти відсутні значення шляхом автоматичного розгляду альтернативних гілок на основі доступних даних.

Недоліки регресії дерева рішень:

- Перенавчання.
- Нестабільність.
- Відсутність безперервності.

Регресія дерева рішень є потужним алгоритмом для прогнозування безперервних числових значень. Його чітка модель, яку можна інтерпретувати, дозволяє легко зрозуміти основні правила та шаблони. Він обробляє як категоричні, так і числові функції, автоматично обробляє відсутні значення та викиди, а також стійкий до нерелевантних функцій. Однак він може бути схильний до переобладнання та чутливості до невеликих варіацій.

Висновки за розділом 1

У розділі 1 були розглянуті регресійні моделі різних типів, здійснений їх порівняльний аналіз, розглянуті переваги та недоліки цих методів. Було обгрунтовано вибір методів для побудови регресійних моделей для розв’язку задачі аналізу та прогнозування злочинності.

РОЗДІЛ 2.
НЕЙРОННІ МЕРЕЖІ. РАДІАЛЬНО-БАЗИСНІ НЕЙРОННІ МЕРЕЖІ.

У завданнях регресії мета полягає в тому, щоб передбачити безперервне числове значення (наприклад, ціну, температуру, оцінку) на основі вхідних даних. Нейронні мережі можна використовувати для регресії шляхом вивчення відображення вхідних функцій на цільовий вихід.

Радиально-базисні функціональні мережі (RBFN) — це клас штучних нейронних мереж, які здобули популярність завдяки своїм унікальним характеристикам і універсальності у вирішенні різних складних проблем. Вони пропонують альтернативу більш традиційним архітектурам нейронних мереж і знайшли застосування у таких сферах, як розпізнавання образів, апроксимація функцій і навіть фінансове моделювання.

2.1 Нейронні мережі

Нейронні мережі (НМ) – це обчислювальні системи, натхнені біологічними нейронними мережами, які складають мозок тварин. НМ ґрунтується на сукупності з'єднаних вузлів, які називають штучними нейронами, а з'єднання між ними називають ребрами. Кожне з'єднання може передавати сигнал до інших нейронів. Після того, як штучний нейрон отримує сигнали, він обробляє їх і може сигналізувати нейронам, з якими його з'єднано. Сигналом у з'єднанні є дійсне число, а вихід кожного нейрона обчислюється деякою нелінійною функцією суми його входів. Нейрони та ребра зазвичай мають вагу, яка підлаштовується в процесі навчання та збільшує або зменшує силу сигналу на з'єднанні. Як правило, нейрони зібрано в шари, і різні шари можуть виконувати різні перетворення даних свого входу. Сигнали проходять від вхідного шару до вихідного, зазвичай, після проходження шарами декілька разів.

Нейронні мережі мають дві основні властивості, що дозволяють вирішувати складні завдання, які сьогодні вважаються важкими для розв’язання. Одним з них є розпаралелювання обробки інформації, а другим – здатність самонавчати, тобто створювати узагальнення. Під терміном узагальнення розуміється здатність отримувати обґрунтований результат спираючись на дані, які зустрічалися у процесі навчання. Однак на практиці при автономній роботі нейронні мережі не можуть забезпечити готових рішень. Їх необхідно інтегрувати у складні системи, або розбити комплексне завдання на послідовність більш простих частин завдання, які можуть вирішуватися нейронними мережами.

Використання нейронних мереж забезпечує такі корисні властивості систем.

1. Нелінійність. Штучні нейрони можуть бути лінійними та нелінійними. Нейронні мережі, побудовані зі сполук нелінійних нейронів, є нелінійними. Нелінійність є надзвичайно важливою властивістю, особливо якщо сам фізичний механізм, який відповідає за формування вхідного сигналу, також є нелінійним.
2. Відображення вхідної інформації у вихідну. Однією з найпопулярніших парадигм навчання є навчання з вчителем. Це означає зміну синаптичних ваг на основі набору маркованих навчальних прикладів. Кожен приклад складається з вхідного сигналу та відповідного йому бажаного відгуку. З цієї множини випадковим чином вибирається приклад, а нейронна мережа модифікує синаптичні ваги для мінімізації розбіжностей бажаного вихідного сигналу і сигналом мережі, що формується відповідно до обраного статистичного критерію. При цьому модифікуються вільні параметри мережі. Приклади, які були використані раніше, можуть бути застосовані знову, але вже в іншому порядку. Це навчання проводиться доти, доки зміни синаптичних ваг не стануть незначними. Таким чином, нейронна мережа навчається на

прикладах, створюючи таблицю відповідностей вхід-вихід для конкретного завдання.

3. Адаптивність. Нейронні мережі мають здатність адаптувати свої синаптичні ваги до змін навколишнього середовища. Зокрема, нейронні мережі, навчені діяти у певному середовищі, можуть бути легко перенавчані для роботи в умовах незначних коливань параметрів середовища. Більше того, для роботи в нестаціонарному середовищі можуть бути створені нейронні мережі, які змінюють свої синаптичні ваги у реальному часі.
4. Контекстна інформація. Знання видаються у структурі нейронної мережі за допомогою її стану активації. Кожен нейрон мережі потенційно може бути схильний до впливу інших її нейронів. Як наслідок існування нейронної мережі безпосередньо пов'язане з контекстною інформацією.
6. Відмовостійкість. Нейронні мережі, у формі електроніки, потенційно стійкі до відмови. Це означає, що за несприятливих умов їхня продуктивність падає незначно. Однак, беручи до уваги розподілений характер зберігання інформації в нейронній мережі, можна стверджувати, що лише серйозні пошкодження структури нейронної мережі суттєво вплинуть на її працездатність.
7. Одноманітність аналізу та проєктування. Нейронні мережі є універсальним механізмом обробки інформації. Це означає, що те саме проєктне рішення нейронної мережі може використовуватися в багатьох предметних областях.

2.2 Особливості РБНМ

Радіально-базисні нейронні мережі (РБНМ) або Мережі радіальної базисної функції (РБФ) — це особливий клас прямої нейронної мережі, що

складається з трьох рівнів: вхідного, прихованого та вихідного. Це принципово відрізняється від більшості архітектур нейронних мереж, які складаються з багатьох рівнів і створюють нелінійність шляхом періодичного застосування нелінійних функцій активації. Вхідний рівень отримує вхідні дані та передає їх у прихований рівень, де відбувається обчислення. Прихований рівень РБНМ є найпотужнішим і сильно відрізняється від більшості нейронних мереж. Вихідний рівень призначений для завдань прогнозування, таких як класифікація або регресія.

РБНМ концептуально подібні до моделей k-найближчих сусідів, хоча реалізація обох моделей сильно відрізняється. Основна ідея радіальних базових функцій полягає в тому, що прогнозоване цільове значення елемента, ймовірно, буде таким же, як і інші елементи з близькими значеннями змінних предикторів. РБНМ розміщує один або багато нейронів РБФ у просторі, що описується змінними предиктора. Простір має кілька вимірів, що відповідають числу присутніх змінних предиктора. Далі обчислюється евклідова відстань від оцінюваної точки до центру кожного нейрона. Радіальна базисна функція, також відома як ядерна функція, застосовується до відстані для обчислення ваги (впливу) кожного нейрона. Назва радіальної базисної функції походить від радіусної відстані, яка є аргументом функції. Чим більша відстань нейрона від точки, що оцінюється, тим менший вплив (вагу) він має.

2.3 Радіально-базисна функція

Радіальна базисна функція — це дійсна функція, значення якої залежить лише від відстані від початку координат. Хоча ми використовуємо різні типи радіальних базисних функцій, найпоширенішою є функція Гауса.

У випадку кількох змінних радіально-базисна нейронна мережа має таку ж кількість вимірів, як і змінні. Якщо три нейрони знаходяться в просторі з двома змінними предикторів, ми можемо передбачити значення за функціями РБФ. Ми можемо обчислити найкраще прогнозоване значення для нової

точки, додавши вихідні значення функцій РБФ, помножені на вагові коефіцієнти, оброблені для кожного нейрона.

Радіальна базова функція для нейрона складається з центру та радіуса (також називається розворотом). Радіус може відрізнятися для різних нейронів. У міру того, як поширення стає більшим, нейрони на відстані від точки мають більший вплив.

2.4 Архітектура РБНМ

Типова архітектура РБНМ складається з вхідного рівня, прихованого рівня та рівня підсумовування.

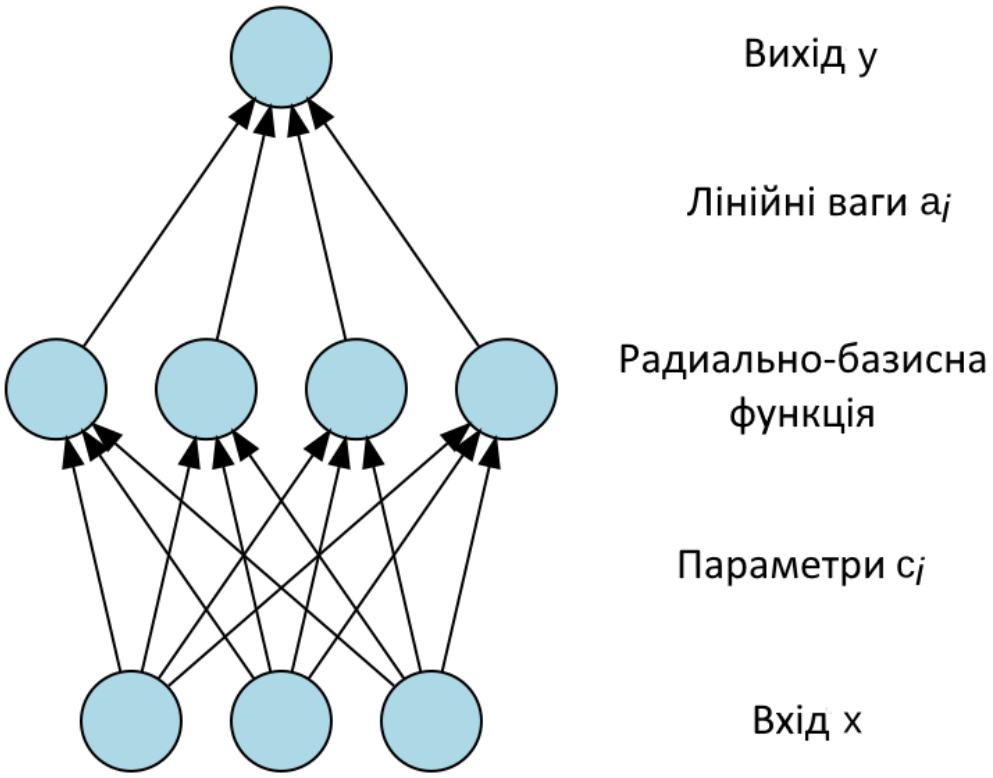


Рис. 2.1 – Схема архітектури РБНМ

Вхідний рівень складається з одного нейрона для кожної змінної предиктора. Вхідні нейрони передають значення кожному нейрону прихованого шару. N-1 нейронів використовуються для вхідних значень, де N позначає кількість зразків. Діапазон значень стандартизований шляхом віднімання медіани та ділення на інтерквартильний діапазон.

Прихований шар містить змінну кількість нейронів (ідеальна кількість визначається процесом навчання). Кожен нейрон містить радіальну базисну функцію з центром у точці. Кількість вимірів збігається з кількістю предикторних змінних. Радіус або поширення РБФ може відрізнятися для кожного виміру.

Коли вектор x вхідних значень подається з вхідного рівня, прихований нейрон обчислює евклідову відстань між тестовим випадком і центральною точкою нейрона. Потім він застосовує функцію ядра, використовуючи значення поширення. Отримане значення надходить до рівня підсумовування.

Значення, отримане з прихованого шару, множиться на вагу, пов’язану з нейроном, і передається до підсумовування. Тут зважені значення складаються, а сума представлена як результат мережі.

Кожен нейрон РБНМ зберігає вектор-прототип (також відомий як центр нейрона) серед векторів навчального набору. Нейрон РБНМ порівнює вхідний вектор із його прототипом і виводить значення від 0 до 1 як міру подібності. Якщо вхідний сигнал такий самий, як і прототип, вихід нейрона дорівнюватиме 1. У міру зростання різниці вхідного та прототипного даних вихідний сигнал експоненціально падає до 0. Форма відповіді нейрона РБНМ — це дзвоноподібна крива. Значення відповіді також називають значенням активації.

На виході мережі знаходиться вихідний вузол, який містить вихідний прогноз для кожного зразка в нашому наборі даних.

2.5 Навчання та переваги РБНМ

Процес навчання мережі RBF з урахуванням обраного типу радіальної базисної функції зводиться до підбору центрів c_i і параметрів σ_i форми базисної функції та до підбору ваги нейронів вихідного шару.

При цьому проблема уточнення ваги нейронів вихідного шару значно спрощується. Позначимо $d = \left[d_1, d_2, \dots, d_p\right]^T$ вектор очікуваних значень, $w = [w_1, w_2, \dots, w_K]^T$ вектор ваги мережі, а G – радіальну матрицю (матриця Гріна). Якщо припустити, що параметри радіальних функцій відомі, то оптимізаційне завдання зводиться до вирішення системи рівнянь, лінійних щодо ваг w [10]

$$G(w) = d \tag{2.1}$$

Оскільки матриця G прямокутна, можна визначити вектор ваг за допомогою операції псевдообернення матриці G, тобто.

$$w = G^+ d \tag{2.2}$$

де $G^+ = (G^T G)^{-1} G^T$ позначає псевдообернення прямокутної матриці G.

Матриця G, що має p рядків та K стовпців, представляє реакції нейронів прихованого шару на чергові збудження векторами x_i (i = 1, 2, …, p). Практично псевдообернення матриці G розраховується з використанням розкладання за власними значеннями, відповідно до яких

$$G = USV^T \tag{2.3}$$

Матриці U і V ортогональні і мають розмірності (pЧp) та (KЧK) відповідно, тоді як S – це псевдодіагональна матриця з розмірністю (pЧK). При цьому K<p, а діагональні елементи $s_1 \geq s_2 \dots \geq s_K \geq 0$. Припустимо, що тільки r перших елементів s_i мають значну величину, а рештою можна знехтувати. Тоді кількість стовпців ортогональних матриць U та V може бути зменшено до r. Отримані таким чином зменшенні матриці U_r та V_r мають вигляд:

$$U_r = [u_1, u_2 \dots u_r],$$
$$V_r = [v_1, v_2 \dots v_r],$$

а матриця $S_r = diag[s_1, s_2 \dots s_r]$, стає повністю діагональною (квадратною). Цю матрицю описує залежність (2.3) у формі

$$G \cong U_r S_r V_r^T. \quad (2.4)$$

Псевдообернена до G матриця визначається в цьому випадку виразом

$$G^+ = V_r S_r^{-1} U_r^T, \quad (2.5)$$

в якому $S_r^{-1} = [1/s_1, 1/s_2 \dots 1/s_r]$, а вектор ваги мережі, що піддається навчанню, задається формулою

$$w = V_r S^{-1} U_r^T d. \quad (2.6)$$

Перевага формули (2.6) – її простота. Вихідні ваги мережі підбираються за один крок простим перемноженням відповідних матриць, причому деякі з них (U_r, V_r) ортогональні і за своєю природою добре впорядковані.

Беручи до уваги рішення (2.6), що визначає значення ваги вихідного шару, головною проблемою навчання радіальних мереж залишається підбір параметрів нелінійних радіальних функцій, особливо центрів c_i.

Одним із найпростіших, хоча й не найефективнішим способом визначення параметрів базисних функцій, вважається випадковий вибір. У цьому рішенні центри c_i базисних функцій вибираються випадковим чином на основі рівномірного розподілу. Такий підхід припустимо стосовно класичних радіальних мереж за умови, що рівномірний розподіл навчальних даних добре відповідає специфіці завдання. При виборі гауссівської форми радіальної функції визначається значення стандартного відхилення s_i, що залежить від розкиду обраних випадковим чином центрів c_i:

$$\varphi(\|x - c_i\|^2) = \exp\left(-\frac{\|x-c_i\|^2}{\frac{d^2}{K}}\right) \quad (2.7)$$

для i = 1, 2, …, K де d позначає максимальну відстань між центрами c_i. З виразу (2.7) виходить, що стандартне відхилення гауссівської функції, яке характеризує ширину кривої, встановлюється при випадковому виборі рівним $\sigma = \frac{d}{\sqrt{2K}}$ і для всіх базисних функцій. Ширина функції пропорційна максимальному розкиду центрів і зменшується зі зростанням їхньої кількості.

Серед багатьох спеціалізованих методів підбору центрів, найбільш важливими є: процес поділу на кластери, що самоорганізується, навчання з учителем і гібридний алгоритм.

Розглянемо переваги мережі РБФ у порівнянні з багатошаровим персептроном. Мережі на основі радіально-базових функцій та багатошаровий персептрон є прикладами нелінійних багатошарових мереж прямого розповсюдження. І ті, й інші є універсальними апроксиматорами. Таким чином, завжди існує мережа РБФ, здатна імітувати багатошаровий персептрон (і навпаки). Однак ці два типи мереж відрізняються за деякими важливими аспектами.

1. Мережі РБФ (у своїй основній формі) мають один прихований шар, у той час як багатошаровий персептрон може мати більше прихованих шарів.
2. Зазвичай обчислювальні (computational) вузли багатошарового персептрона, розташовані у прихованих та вихідному шарах, мають одну й ту саму модель нейрона. З іншого боку, обчислювальні вузли прихованого шару мережі РБФ можуть докорінно відрізнятися від вузлів вихідного шару і служити різним цілям.
3. Прихований шар у мережах РБФ є нелінійним, тоді як вихідний – лінійним. У той же час приховані та вихідний шари багатошарового персептрона, що використовується як класифікатор, є нелінійними. Якщо багатошаровий персептрон використовується для вирішення задач нелінійної регресії, як вузли вихідного шару зазвичай вибираються лінійні нейрони.
4. Аргумент функції активації кожного прихованого вузла мережі РБФ є Евклідовою нормою (відстань) між вхідним вектором і центром радіальної активації. У той же час аргумент функції активації кожного прихованого вузла багатошарового персептрона - це скалярний добуток вхідного вектора і синаптичних векторів ваг даного нейрона.

5. Багатошаровий персептрон забезпечує глобальну апроксимацію нелінійного відображення. З іншого боку, мережа РБФ за допомогою локалізованих нелінійностей, що експоненційно зменшуються (тобто функцій Гаусса) створює локальну апроксимацію нелінійного відображення.

Це, у свою чергу, означає, що для апроксимації нелінійного відображення за допомогою багатошарового персептрона може знадобитися менше параметрів, ніж для мережі РБФ при однаковій точності обчислень.

Ще одна важлива перевага радіальних мереж – значно спрощений алгоритм навчання. За наявності лише одного прихованого шару та тісного зв'язку активності нейрона з відповідною областю простору навчальних даних точка початку навчання виявляється набагато ближче до оптимального рішення, ніж це має місце у багатошарових мережах. Крім того, можна відокремити етап підбору параметрів базисної функції від підбору значень ваги мережі (гібридний алгоритм), що сильно спрощує і прискорює процес навчання. Виграш у часі стає ще більшим, якщо взяти до уваги процедуру формування оптимальної (з погляду здатності до узагальнення) структури мережі. При використанні багатошарових мереж це дуже трудомістка задача, яка вимагає, як правило, багаторазового повторення навчання або донавчання. Для радіальних мереж, особливо заснованих на ортогоналізації, формування оптимальної структури мережі виявляється природним етапом процесу навчання, що не потребує додаткових зусиль.

Лінійні характеристики вихідного шару мережі РБФ означають, що така мережа тісно пов'язана з персептроном Розенблатта, ніж багатошаровим персептроном. Тим не менш, мережі РБФ відрізняються від цього персептрона тим, що здатні виконувати нелінійні перетворення вхідного простору. Це було добре продемонстровано на прикладі рішення задачі XOR, яка не може бути вирішена жодним лінійним персептроном, але легко вирішується мережею РБФ.

Висновки за розділом 2

У розділі 2 були розглянуті неронні мережі та РБНМ, їх переваги та недоліки. РБНМ можуть стати гарним кандидатом для аналізу та прогнозування злочинності, особливо при роботі з великими наборами даних.

РОЗДІЛ 3.
АНАЛІЗ ТА ПРОГНОЗУВАННЯ ЗЛОЧИННОСТІ НА ДАТАСЕТІ COMMUNITIES IN THE US

3.1 Опис датасету

Датасет «Communities in the US» містить у собі соціально-економічні дані з перепису 1990 року, дані правоохоронних органів з опитування щодо управління правоохоронними органами та адміністративної статистики 1990 року та дані злочинності з ФБР 1995 року. Цей набір даних містить 147 атрибутів і 2216 екземплярів і також має у собі пропущені значення. Розглянемо цільові атрибути даного датасету:

Таблиця 3.1

Цільові атрибути Communities in the US

Назва змінної	Тип	Пропущені змінні	Опис
Murders	Integer	ні	Кількість вбивств у 1995 р.
murdPerPop	Continuous	ні	Кількість вбивств на 100 тис. населення
Rapes	Integer	так	Кількість зґвалтувань у 1995 р.
rapesPerPop	Continuous	так	Кількість зґвалтувань на 100 тис. населення
Robberies	Integer	так	Кількість пограбувань у 1995 р.
robbbPerPop	Continuous	так	Кількість пограбувань на 100 тис. населення
Assaults	Integer	так	Кількість нападів у 1995 році

assaultPerPop	Continuous	так	Кількість нападів на 100 тис. населення
Burglaries	Integer	так	Кількість крадіжок зі зломом у 1995 році
burglPerPop	Continuous	так	Кількість крадіжок зі зломом на 100 тис. населення
Larcenies	Integer	так	Кількість крадіжок у 1995 році
larcPerPop	Continuous	так	Кількість крадіжок на 100 тис. населення
autoTheft	Integer	ні	Кількість крадіжок автомобілів у 1995 році
autoTheftPerPop	Continuous	ні	Кількість крадіжок автомобілів на 100 тис. населення
Arsons	Integer	так	Кількість підпалів у 1995 р.
arsonsPerPop	Continuous	так	Кількість підпалів на 100 тис. населення
violentPerPop	Continuous	так	Загальна кількість насильницьких злочинів на 100 тис. населення
nonViolPerPop	Continuous	так	Загальна кількість ненасильницьких злочинів на 100 тис. населення

3.2 Попередня обробка даних

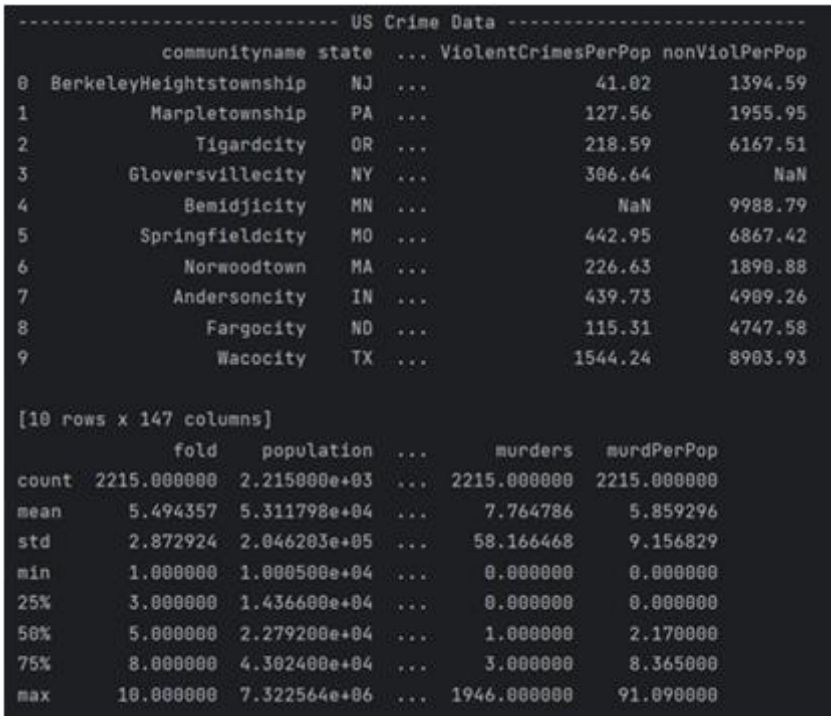
Так як датасет містить порожні значення, то потрібно їх замінити на інші значення. Цей процес називається імпутацією. Нехай усі пропущені значення будуть замінені медіанним значенням кожного зі стовпців. Медіана – це

значення, яке є серединою певного набору і поділяє його навпіл. Розглянемо код для цього:

```
[3] print("---------------------------- US Crime Data ----------------------------")
    print(uscrime_df.head(10))
    print(uscrime_df.describe())
    print("NaN values:")
    print(uscrime_df.isnull().sum())
    print("Total amount of NaN values: %s" % uscrime_df.isnull().sum().sum())

    Show hidden output

[4] #Removing non-predictive features
    uscrime_df = uscrime_df.drop(columns=['state','countyCode','communityCode','communityname','fold'], axis=1)
    #Filling all NaN with median values
    feat_miss = uscrime_df.columns[uscrime_df.isnull().any()]
    uscrime_df[uscrime_df.columns] = uscrime_df[uscrime_df.columns].apply(pd.to_numeric, errors='coerce')
    uscrime_df.fillna(uscrime_df[feat_miss].median(),inplace=True)

[5] #The new dataset
    print("\n-------------------- US Crime Data (formated) ----------------------")
    print(uscrime_df.head(10))
    print(uscrime_df.describe())
    print("NaN values:")
    print(uscrime_df.isnull().sum())
    print("Total amount of NaN values: %s" % uscrime_df.isnull().sum().sum())

    Show hidden output
```

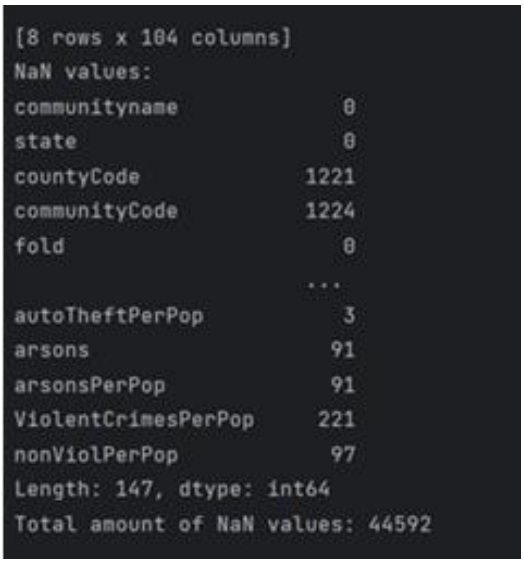
Рис. 3.1 – Код для завантаження і обробки датасету

Тут датасет завантажується через функцію read_csv бібліотеки pandas, після чого він перетворюється на датафрейм функцією DataFrame і замінює усі значення «?» на NaN. Після цього виводяться перші 10 рядків датасету, інформацію про його змінні та кількість пустих значень:



```
---------------------------- US Crime Data ----------------------------
            communityname state  ... ViolentCrimesPerPop nonViolPerPop
0  BerkeleyHeightstownship    NJ  ...              41.02       1394.59
1          Marpletownship    PA  ...             127.56       1955.95
2              Tigardcity    OR  ...             218.59       6167.51
3        Gloversvillecity    NY  ...             306.64           NaN
4             Bemidjicity    MN  ...                NaN       9988.79
5         Springfieldcity    MO  ...             442.95       6867.42
6            Norwoodtown     MA  ...             226.63       1890.88
7            Andersoncity    IN  ...             439.73       4909.26
8               Fargocity    ND  ...             115.31       4747.58
9                Wacocity    TX  ...            1544.24       8903.93

[10 rows x 147 columns]
             fold    population  ...      murders   murdPerPop
count  2215.000000  2.215000e+03  ...  2215.000000  2215.000000
mean      5.494357  5.311798e+04  ...     7.764786     5.859296
std       2.872924  2.046203e+05  ...    58.166468     9.156829
min       1.000000  1.000500e+04  ...     0.000000     0.000000
25%       3.000000  1.436600e+04  ...     0.000000     0.000000
50%       5.000000  2.279200e+04  ...     1.000000     2.170000
75%       8.000000  4.302400e+04  ...     3.000000     8.365000
max      10.000000  7.322564e+06  ...  1946.000000    91.090000
```

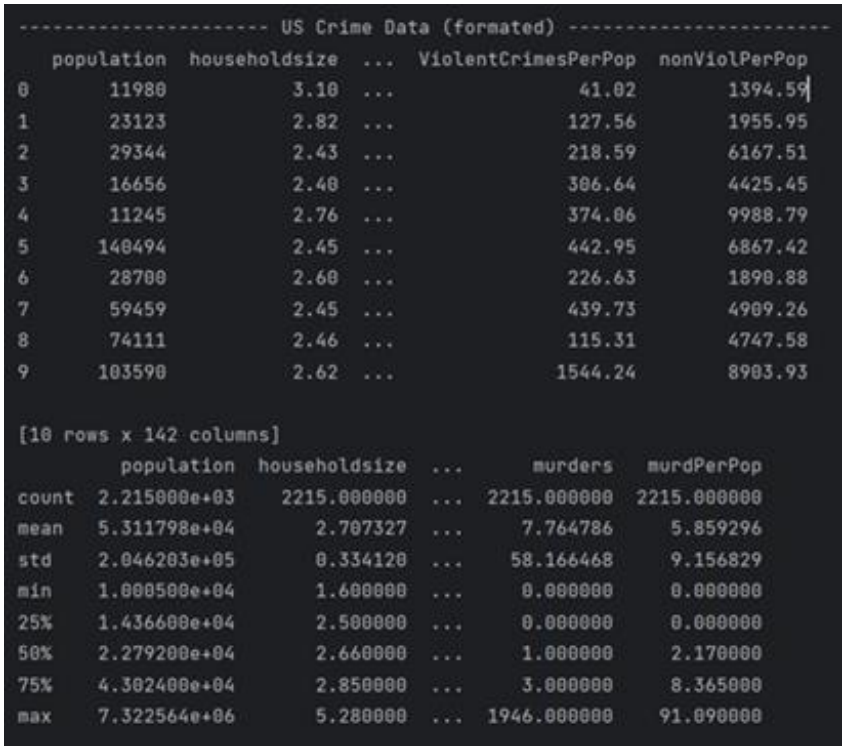
Рис. 3.2 – Початкова інформація про датасет



```
[8 rows x 104 columns]
NaN values:
communityname            0
state                    0
countyCode            1221
communityCode         1224
fold                     0
                       ...
autoTheftPerPop          3
arsons                  91
arsonsPerPop            91
ViolentCrimesPerPop    221
nonViolPerPop           97
Length: 147, dtype: int64
Total amount of NaN values: 44592
```

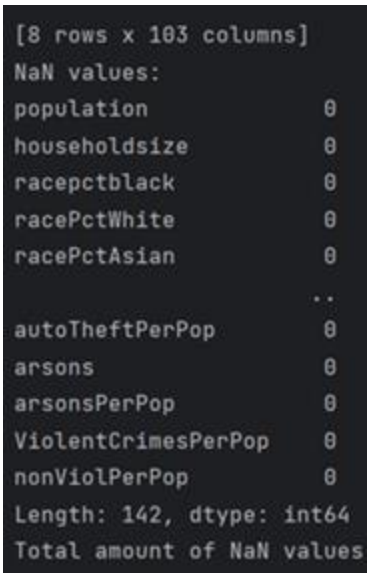
Рис. 3.3 – Початкова кількість пропущених значень датасету

Далі виконується видалення нечислових стовпців та імпутація, і заново виводяться перші 10 рядків датасету, інформацію про його змінні та кількість пустих значень:



```
---------------------- US Crime Data (formated) ----------------------
   population  householdsize  ...  ViolentCrimesPerPop  nonViolPerPop
0       11980           3.10  ...                41.02        1394.59
1       23123           2.82  ...               127.56        1955.95
2       29344           2.43  ...               218.59        6167.51
3       16656           2.40  ...               306.64        4425.45
4       11245           2.76  ...               374.06        9988.79
5      140494           2.45  ...               442.95        6867.42
6       28700           2.60  ...               226.63        1890.88
7       59459           2.45  ...               439.73        4909.26
8       74111           2.46  ...               115.31        4747.58
9      103590           2.62  ...              1544.24        8903.93

[10 rows x 142 columns]
         population  householdsize  ...      murders   murdPerPop
count  2.215000e+03    2215.000000  ...  2215.000000  2215.000000
mean   5.311798e+04       2.707327  ...     7.764786     5.859296
std    2.046203e+05       0.334120  ...    58.166468     9.156829
min    1.000500e+04       1.600000  ...     0.000000     0.000000
25%    1.436600e+04       2.500000  ...     0.000000     0.000000
50%    2.279200e+04       2.660000  ...     1.000000     2.170000
75%    4.302400e+04       2.850000  ...     3.000000     8.365000
max    7.322564e+06       5.280000  ...  1946.000000    91.090000
```

Рис. 3.4 – Інформація про датасет після імпутації



```
[8 rows x 103 columns]
NaN values:
population              0
householdsize           0
racepctblack            0
racePctWhite            0
racePctAsian            0
                       ..
autoTheftPerPop         0
arsons                  0
arsonsPerPop            0
ViolentCrimesPerPop     0
nonViolPerPop           0
Length: 142, dtype: int64
Total amount of NaN values
```

Рис. 3.5 – Кількість пропущених значень датасету після імпутації

3.3 Побудова моделей

Таким чином, датасет тепер має лише числові значення, і його можна використовувати для побудови моделей. Однак перед вибором тренувальних

даних, розглянемо кореляцію між цільовими змінними та кількома першими змінними:

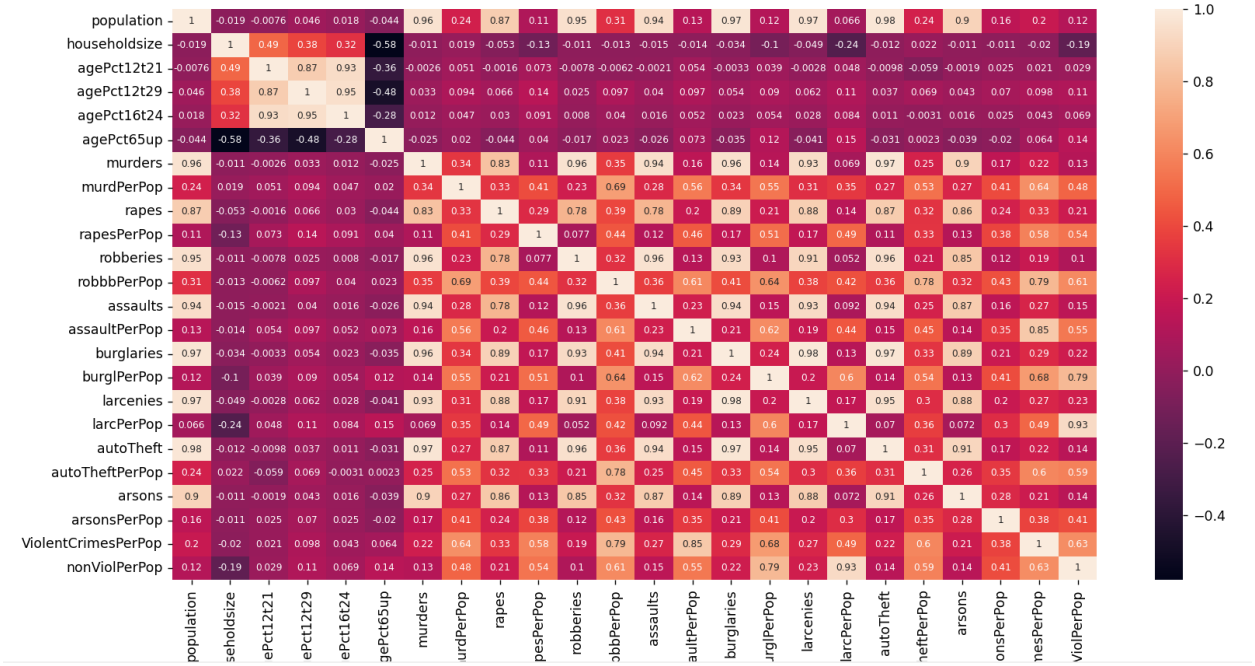


Рис. 3.6 – Матриця кореляції цільових значень і перших семи стовпців датасету

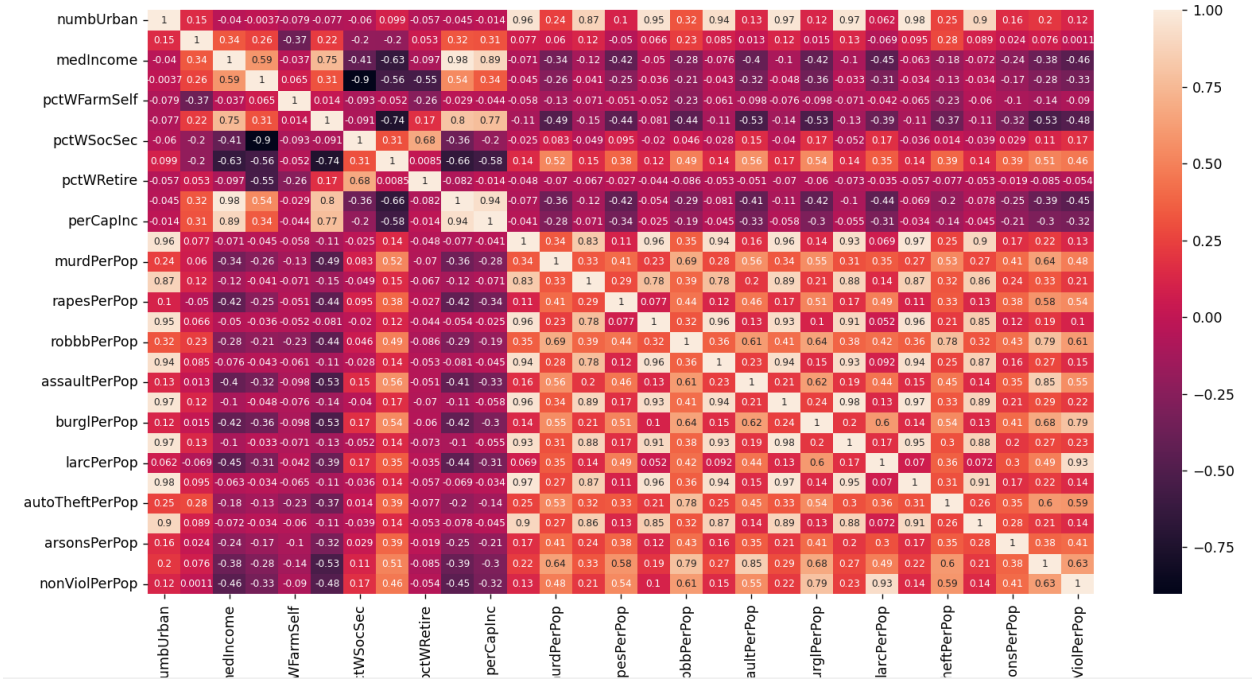


Рис. 3.7 – Матриця кореляції цільових значень і статусів доходів

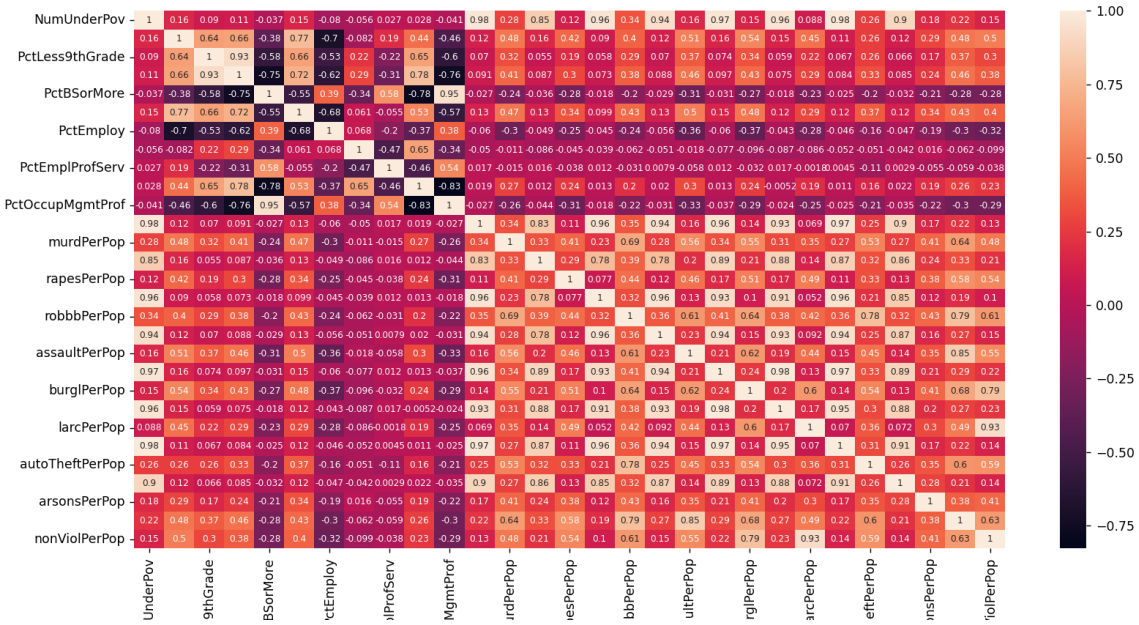


Рис. 3.8 – Матриця кореляції цільових значень і статусів роботи

Дані матриці кореляції були побудовані наступним кодом:

```
X_train1 = uscrime_df[['population','householdsize', 'agePct12t21','agePct12t29','agePct16t24','agePct65up',
                       'murders','murdPerPop','rapes','rapesPerPop','robberies','robbbPerPop',
                       'assaults','assaultPerPop','burglaries','burglPerPop','larcenies','larcPerPop',
                       'autoTheft','autoTheftPerPop','arsons','arsonsPerPop','ViolentCrimesPerPop','nonViolPerPop']]
X_train2 = uscrime_df[['numbUrban','pctUrban', 'medIncome','pctWWage','pctWFarmSelf','pctWInvInc', 'pctWSocSec',
                       'pctWPubAsst','pctWRetire','medFamInc','perCapInc',
                       'murders','murdPerPop','rapes','rapesPerPop','robberies','robbbPerPop',
                       'assaults','assaultPerPop','burglaries','burglPerPop','larcenies','larcPerPop',
                       'autoTheft','autoTheftPerPop','arsons','arsonsPerPop','ViolentCrimesPerPop','nonViolPerPop']]
X_train3 = uscrime_df[['NumUnderPov','PctPopUnderPov','PctLess9thGrade','PctNotHSGrad','PctBSorMore','PctUnemployed',
                       'PctEmploy','PctEmplManu','PctEmplProfServ','PctOccupManu','PctOccupMgmtProf',
                       'murders','murdPerPop','rapes','rapesPerPop','robberies','robbbPerPop',
                       'assaults','assaultPerPop','burglaries','burglPerPop','larcenies','larcPerPop',
                       'autoTheft','autoTheftPerPop','arsons','arsonsPerPop','ViolentCrimesPerPop','nonViolPerPop']]

for x in [X_train1, X_train2, X_train3]:
    corr_matrix = x.corr(method='pearson')
    plt.figure()
    sns.heatmap(corr_matrix, annot=True, annot_kws={'size': 7})
    plt.show()
```

Рис. 3.9 – Код для побудови матриць кореляції

Так як датасет містить у собі понад 100 атрибутів, для побудови моделей будемо використовувати лише частину них. Нехай для *X* будуть атрибути, які

мали значний вплив на кореляції між цільовими значеннями, а для Y – цільовий атрибут про кількість вбивств:

```
X_train = uscrime_df[['population','agePct12t29','numbUrban','pctUrban','medIncome','pctWInvInc',
                      'pctWSocSec','pctWPubAsst','medFamInc','NumUnderPov','PctPopUnderPov',
                     'PctLess9thGrade','PctNotHSGrad','PctBSorMore','PctUnemployed',
                      'PctEmploy','PctOccupManu','PctOccupMgmtProf']]

target = uscrime_df[['murders']]
```

Рис. 3.10 – Код для розподілу атрибутів

Нормалізуємо значення X і розділяємо дані на тренувальні та тестові (при цьому з усіх значень X 70% – тренувальні дані, а 30% – тестові) наступними рядками коду:

```
std_scaler = StandardScaler()
scaled_X_train = std_scaler.fit_transform(X_train)
X_train_scaled = pd.DataFrame(scaled_X_train)

xtrain, xtest, ytrain, ytest = train_test_split( *arrays: X_train_scaled, target, test_size=0.3, random_state=5)
```

Рис. 3.11 – Код для нормалізації та розподілу даних на тренувальні та тестові набори

Тепер створимо кілька моделей регресії з параметрами за замовчуванням. Почнемо з простої лінійної регресії. Розглянемо код для створення цієї моделі (рис. 3.12) і результат його виконання (рис. 3.13):

```
print("\n===Linear Regression (default parameters):===")
linear_regression = LinearRegression().fit(xtrain, ytrain)
print("Training set score:", round(linear_regression.score(xtrain, ytrain), 3))
print("Testing set score:", round(linear_regression.score(xtest,ytest),3))
print("Coefficients :", linear_regression.coef_)
print("Intercept:", linear_regression.intercept_)
```

Рис. 3.12 – Код створення моделі лінійної регресії за замочуванням

```
===Linear Regression (default parameters):===
Training set score: 0.96
Testing set score: 0.93
Coefficients : [[-2.70086723e+01  5.07000158e-02  3.05191991e+01 -2.64671703e+00
   3.71649105e+00 -1.74376444e+00  4.42692121e-01  2.98511915e-01
  -3.50677883e+00  5.33358459e+01 -3.57082684e+00 -4.64066661e+00
   6.79808282e+00  2.39373799e+00 -1.41968279e-01  1.22865382e-01
  -2.16397781e-01  1.13199816e+00]]
Intercept: [7.7202764]
```

Рис. 3.13 – Результат лінійної регресії за замочуванням

З результатів можна побачити, що отримано точність 0,96 для тренувального набору і 0,93 для тестового набору. З коефіцієнтів також можна побачити, які вхідні дані мають більший вплив на результат (так наприклад у цій моделі, особи з 12 до 29 років за межею бідності і без повної середньої освіти мають більший шанс на вчинення вбивства). Зміщення (intercept) дорівнює 7,7, тобто якщо всі показники дорівнюють 0, то модель видаватиме значення 7,7.

Дана модель видає непогані результати, але важко сказати, чи є вона перенавченою, чи ні.

Розглянемо інші моделі регресії. Нижче наведено код для створення моделі гребневої регресії та результат його виконання (рис. 3.15):

```
print("\n===Ridge Regression (default parameters):===")
ridge_regression = Ridge().fit(xtrain, ytrain)
print("Training set score:", round(ridge_regression.score(xtrain, ytrain), 3))
print("Testing set score:", round(ridge_regression.score(xtest,ytest),3))
print("Coefficients :", ridge_regression.coef_)
print("Intercept:", ridge_regression.intercept_)
```

Рис. 3.14 – Код для створення моделі гребневої регресії за замочуванням

```
===Ridge Regression (default parameters):===
Training set score: 0.959
Testing set score: 0.927
Coefficients : [[-4.24796638e+00 -1.93376147e-02  9.68115483e+00 -1.81387754e+00
   3.57899003e+00 -1.71141813e+00  4.63573300e-01  2.29176412e-01
  -3.30073980e+00  5.12447412e+01 -3.35546712e+00 -4.46886507e+00
   6.60866206e+00  2.25478797e+00 -1.28404619e-01  1.24639949e-01
  -8.84993554e-02  1.21823013e+00]]
Intercept: [7.70852326]
```

Рис. 3.15 – Результат гребневої регресії за замочуванням

З результатів можна побачити, що отримано точність 0,959 для тренувального набору і 0,927 для тестового набору, що майже повністю збігається зі значеннями для лінійної регресії. З коефіцієнтів також можна побачити, які вхідні дані мають більший вплив на результат (так наприклад у цій моделі, особи за межею бідності, які проживають у місті і без повної середньої освіти мають більший шанс на вчинення вбивства). Зміщення (intercept) дорівнює 7,7, тобто якщо всі показники дорівнюють 0, то модель видаватиме значення 7,7.

Дана модель також видає непогані результати, але проблема, яка була присутня моделі лінійній регресії залишається незмінною.

Далі розглянемо моделі регресії ласо. Нижче наведено код для створення моделі та результат його виконання (рис. 3.17):

```
print("\n===Lasso Regression (default parameters):===")
lasso_regression = Lasso(max_iter=10000).fit(xtrain, ytrain)
print("Training set score:", round(lasso_regression.score(xtrain, ytrain), 3))
print("Testing set score:", round(lasso_regression.score(xtest,ytest),3))
print("Coefficients :", lasso_regression.coef_)
print("Intercept:", lasso_regression.intercept_)
```

Рис. 3.16 – Код для побудови моделі регресії ласо за замочуванням

```
===Lasso Regression (default parameters):===
Training set score: 0.958
Testing set score: 0.923
Coefficients : [ 7.01660361 -0.11801716  0.         -0.          0.          0.
  0.         -0.          0.         48.61842725 -0.50611236 -0.
 -0.          0.         -0.          0.         -0.          0.        ]
Intercept: [7.72192759]
```

Рис. 3.17 – Результат регресії ласо за замочуванням

З результатів можна побачити, що отримано точність 0,958 для тренувального набору і 0,923 для тестового набору, що майже повністю збігається зі значеннями для лінійної та гребневої регресії. З коефіцієнтів також можна побачити, які вхідні дані мають більший вплив на результат (так наприклад у цій моделі, особи за межею бідності, які проживають у населених місцях мають більший шанс на вчинення вбивства). Зміщення (intercept) дорівнює 7,7, тобто якщо всі показники дорівнюють 0, то модель видаватиме значення 7,7.

Дана модель також видає непогані результати, але проблема, яка була присутня у попередніх моделях залишається незмінною.

Тепер розглянемо модель регресії К-найближчих сусідів. Нижче наведено код для створення цієї моделі та результат його виконання (рис. 3.19):

```
print("\n===Knn regression (default parameters):===")
knn_model = KNeighborsRegressor().fit(xtrain, ytrain)
print("Training set score:")
print(round(knn_model.score(xtrain, ytrain), 3))
print("Testing set score:")
print(round(knn_model.score(xtest, ytest), 3))
```

Рис. 3.18 – Код для побудови моделі регресії К-найближчих сусідів за замочуванням

```
===Knn regression (default parameters):===
Training set score:
0.705
Testing set score:
0.74
```

Рис. 3.19 – Результат регресії К-найближчих сусідів за замочуванням

З результатів можна побачити, що отримано точність 0,705 для тренувального набору і 0,74 для тестового набору. Незважаючи на порівняно з попередніми моделями низьку точність, такі результати є більш реалістичними.

Далі розглянемо модель регресії дерева рішень. Нижче наведено код для створення цієї моделі та результат його виконання (рис. 3.21):

```
print("\n===Decision Tree Regression (default parameters):===")
decision_tree_regression = DecisionTreeRegressor().fit(xtrain, ytrain)
print("Training set score:", round(decision_tree_regression.score(xtrain, ytrain), 3))
print("Testing set score:", round(decision_tree_regression.score(xtest, ytest), 3))
```

Рис. 3.20 – Код для побудови моделі регресії дерева рішень за замочуванням

```
===Decision Tree Regression (default parameters):===
Training set score: 1.0
Testing set score: 0.871
```

Рис. 3.21 – Результат регресії дерева рішень за замочуванням

З результатів можна побачити, що отримано точність 1,0 для тренувального набору і 0,871 для тестового набору. Результати на тренувальному наборі можуть свідчити про перенавчання моделі, хоча на тестових точність більш менш реалістична.

Наостанок побудуємо радіально-базисну нейронну мережу. Код програми і результат виконання наступний:

```
from keras.layers import Layer
import tensorflow.keras.backend as K
import tensorflow as tf


class RBFLayer(Layer):
    def __init__(self, units, gamma, **kwargs):
        super(RBFLayer, self).__init__(**kwargs)
        self.units = units
        self.gamma = tf.cast(gamma, dtype=K.floatx())

    def build(self, input_shape):
        self.mu = self.add_weight(name='mu',
                                  shape=(int(input_shape[1]), self.units),
                                  initializer='uniform',
                                  trainable=True)
        super(RBFLayer, self).build(input_shape)

    def call(self, inputs):
        diff = tf.expand_dims(inputs,-1) - self.mu
        l2 = K.sum(K.pow(diff, 2), axis=1)
        res = K.exp(-1 * self.gamma * l2)
        return res

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.units)


from keras.layers import Dense, Flatten
from keras.models import Sequential
from keras.losses import binary_crossentropy

model = Sequential()
model.add(Flatten(input_shape=(18, )))
model.add(RBFLayer(18, 0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss=binary_crossentropy)
```

Рис. 3.21 – Код для побудови РБНМ

```
model.fit(xtrain, ytrain, batch_size=256, epochs=8)

Epoch 1/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.5319
Epoch 2/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4064
Epoch 3/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.2999
Epoch 4/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2491
Epoch 5/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2034
Epoch 6/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1600
Epoch 7/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0173
Epoch 8/8
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: -0.0615
<keras.src.callbacks.history.History at 0x7d02d79a1540>

test_loss = model.evaluate(xtest, ytest)
print("test loss", test_loss)

21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0440
test loss 0.033198896795511246
```

Рис. 3.22 – Результат РБНМ

Отримана похибка є непоганою, і свідчить про те, що даний метод доцільно використовувати для аналізу та прогнозування даних.

Таким чином були побудовані моделі регресії, усі які показують точність більшою за 0,7, але меншу за 0,95 на тестових наборах. Серед цих моделей більш точними є моделі регресії ласо, гребневої регресії та лінійної регресії.

3.4 Крос-валідація для моделей регресії

Побудуємо кілька моделей за допомогою *GridSearchCV*, яка буде шукати найкращий параметр для моделі шляхом крос-валідації. Не всі згадані вище моделі мать параметри, якими можна керувати, тому розглянемо лише моделі регресії К-найближчих сусідів (параметр 'n_neighbors'), гребневої

регресії (параметр 'alpha') та регресії ласо (параметр 'alpha'). Код програми і результат виконання (рис. 3.24) наведено нижче:

```
print("\n===Parameters Cross-Validation===")
parameters_knn = {'n_neighbors': [2, 3, 4, 5, 6, 7, 8, 9, 10]}
grid_search_knn = GridSearchCV(knn_model, parameters_knn, n_jobs=-1, verbose=True).fit(xtrain, ytrain)
print("Knn best parameters:", grid_search_knn.best_params_)
print("Knn best score:", grid_search_knn.best_score_, "\n")

parameters_ridge = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10]}
grid_search_ridge = GridSearchCV(ridge_regression, parameters_ridge, n_jobs=-1, verbose=True).fit(xtrain, ytrain)
print("Ridge best parameters:", grid_search_ridge.best_params_)
print("Ridge best score:", grid_search_ridge.best_score_, "\n")

parameters_lasso = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 15, 20, 25]}
grid_search_lasso = GridSearchCV(lasso_regression, parameters_lasso, n_jobs=-1, verbose=True).fit(xtrain, ytrain)
print("Lasso best parameters:", grid_search_lasso.best_params_)
print("Lasso best score:", grid_search_lasso.best_score_)
```

Рис. 3.23 – Код для крос-валідації моделей регресії

```
===Parameters Cross-Validation===
Fitting 5 folds for each of 9 candidates, totalling 45 fits
Knn best parameters: {'n_neighbors': 2}
Knn best score: 0.7008605973466026

Fitting 5 folds for each of 9 candidates, totalling 45 fits
Ridge best parameters: {'alpha': 0.001}
Ridge best score: 0.9233320688115004
Lasso best parameters: {'alpha': 0.1}
Lasso best score: 0.9251836460848628
```

Рис. 3.24 – Крос-валідація для моделей регресії

Бачимо, що отримано наступні найкращі параметри для моделей:

- Регресія К-наближчих сусідів — n_neighbors = 2
- Регресія ласо — alpha = 0,001
- Гребнева регресія — alpha = 0,1

Підставимо ці параметри у моделі. Код програми:

```
print("\n===Ridge Regression (best parameters):===")
ridge_regression_best = Ridge(alpha=0.001).fit(xtrain, ytrain)
print("Training set score:", round(ridge_regression_best.score(xtrain, ytrain), 3))
print("Testing set score:", round(ridge_regression_best.score(xtest, ytest), 3))
print("Coefficients :", ridge_regression_best.coef_)
print("Intercept:", ridge_regression_best.intercept_)

print("\n===Lasso Regression (best parameters):===")
lasso_regression_best = Lasso(alpha=0.1, max_iter=10000).fit(xtrain, ytrain)
print("Training set score:", round(lasso_regression_best.score(xtrain, ytrain), 3))
print("Testing set score:", round(lasso_regression_best.score(xtest, ytest), 3))
print("Coefficients :", lasso_regression_best.coef_)
print("Intercept:", lasso_regression_best.intercept_)

print("\n===Knn regression (best parameters):===")
knn_model = KNeighborsRegressor(n_neighbors=2).fit(xtrain, ytrain)
print("Training set score:")
print(round(knn_model.score(xtrain, ytrain), 3))
print("Testing set score:")
print(round(knn_model.score(xtest, ytest), 3))
```

Рис. 3.25 – Код для побудови моделей гребневої регресії, регресії ласо та регресії КНН з найкращими параметрами

Отримуємо наступні результати:

```
===Ridge Regression (best parameters):===
Training set score: 0.96
Testing set score: 0.93
Coefficients : [[-2.69198770e+01  5.05110126e-02  3.04325029e+01 -2.64323266e+00
   3.71630085e+00 -1.74376290e+00  4.42863460e-01  2.98214982e-01
  -3.50642647e+00  5.33332465e+01 -3.57043995e+00 -4.64017226e+00
   6.79758220e+00  2.39349304e+00 -1.41940863e-01  1.22931564e-01
  -2.15981212e-01  1.13221056e+00]]
Intercept: [7.72023584]
```

Рис. 3.26 – Результат гребневої регресії з найкращими параметрами

```
===Lasso Regression (best parameters):===
Training set score: 0.959
Testing set score: 0.93
Coefficients : [ 0.         -0.3174724   3.70896898 -1.24988479  0.60001384 -1.09562463
  0.          0.          0.         52.94556115 -2.545345   -2.04794283
  3.1684749   0.38655968 -0.         -0.          0.          1.01540952]
Intercept: [7.7213012]
```

Рис. 3.27 – Результат регресії ласо з найкращими параметрами

```
===Knn regression (best parameters):===
Training set score:
0.934
Testing set score:
0.832
```

Рис. 3.28 – Результат регресії К-середніх з найкращими параметрами

З отриманих результатів можна побачити, що крос-валідація підвищила точність усіх моделей, особливо для моделі К-найближчих сусідів.

Розглянемо табл. 3.2, у якій наведено результати роботи всіх моделей:

Таблиця 3.2

Результати роботи моделей

Модель	**Точність на тренувальному наборі**	**Точність на тестовому наборі**	**Зміщення (Intercept)**
Linear Regression (default)	0,96	0,93	7,72
Ridge Regression (default)	0,959	0,927	7,72
Ridge Regression (best)	0,96	0,93	7,72
Lasso Regression (default)	0,953	0,928	7,72
Lasso Regression (best)	0,959	0,93	7,72
K-nn Regression (default)	0,705	0,74	-
K-nn Regression (best)	0,934	0,832	-
Decision Tree Regression (default)	1,0	0,871	-
Модель	**Похибка на тренувальному наборі**	**Похибка на тестовому наборі**	**Зміщення (Intercept)**
РБНМ	-0,06	0,03	**-**

Висновки за розділом 3

У Розділі 3 був опрацьований набір даних Communities and Crime Unnormalized, на якому, після попередньої обробки, були навчані 9 регресійних моделей.

ВИСНОВКИ

Кваліфікаційна робота присвячена актуальній задачі довгострокового та оперативного аналізу статистичної інформації з подальшим прогнозуванням факторів та чинників, які впливають на показники злочинності. В результаті дослідження розроблено та оцінено кілька моделей машинного навчання для аналізу та прогнозування злочинності. Задля побудови моделі аналізу впливу на злочинність соціальних, демографічних чинників були розв’язані наступні задачі:

- сформульована задача кваліфікаційної роботи, мету, об’єкт та предмет дослідження;
- проведений огляд наукових публікацій щодо використання методів машинного навчання для аналізу злочинності;
- проведений огляд регресійних методів машинного навчання, зокрема лінійна регресія, дерева рішень, випадкові ліси, алгоритм k-найближчих сусідів та нейронні мережі для аналізу та прогнозування злочинності, визначені їх переваги та недоліки, обґрунтований вибір радіально-базисних нейронних мереж для побудови моделі прогнозування.
- розроблено програмно-алгоритмічну реалізацію розв’язку задачі аналізу та прогнозування злочинності використовуючи мову програмування Python
- виконано тестування моделі на наборі даних «Communities in the US», який містить у собі соціально-економічні показники і з перепису 1990 року, дані правоохоронних органів з опитування щодо управління правоохоронними органами та адміністративної статистики 1990 року та дані злочинності з ФБР 1995 року, аналіз та інтерпретацію отриманих результатів;

- проведено дослідження щодо вибору кращої моделі серед 9 в термінах зменшення похибки прогнозування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «Crime Analysis Through Machine Learning» / S. Kim, P. Joshi, P. S. Kalsi та P. Taheri, 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). Канада: Ванкувер, BC, 2018. С. 415-420. doi: 10.1109/IEMCON.2018.8614828.
2. «Crime Prediction Using Machine Learning and Deep Learning: A Systematic Review and Future Directions» / V. Mandalapu, L. Elluri, P. Vyas, N. Roy. IEEE Access. 2023. Vol. 11. С. 60153–60170.
3. Walczak S. «Predicting Crime and Other Uses of Neural Networks in Police Decision Making». *Frontiers in Psychology* Vol. 12. 2021.
4. Tom M. Mitchell. «Machine Learning». McGraw-Hill Science/Engineering/Math, 1997. 414 с.
5. «Python: Deeper Insights into Machine Learning» / Sebastian Raschka, David Julian, John Hearty. Packt Publishing Ltd. Livery Place 35 Livery Street Birmingham B3 2PB, UK, 2016. 901 с. - ISBN 978-1-78712-857-6
6. Michael Bowles. «Machine Learning in Python: Essential Techniques for Predictive Analysis». United States: Wiley. 2015. 360 с.
7. Analyzing UCI Crime and Communities Dataset [Електронний ресурс], URL: https://www.kaggle.com/code/kkanda/analyzing-uci-crime-and-communities-dataset. (Дата звернення: 01.10.2024).
8. David Kriesel, 2007, «A Brief Introduction to Neural Networks», available at http://www.dkriesel.com.
9. What are Radial Basis Functions Neural Networks? Everything You Need to Know. [Електронний ресурс], URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-are-radial-basis-functions-neural-networks. (Дата звернення: 02.12.2024).
10. Simon Haykin. «Neural Networks: A Comprehensive Foundation 2Nd Ed.» Prentice-Hall Of India Pvt. Limited. 1999. ISBN: 9788120323735

11. Stanisław Osowski. «Sieci neuronowe w ujęciu algorytmicznym». Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 1996. ISBN 83-204-2197-7
12. Regression algorithms. [Електронний ресурс], URL: https://arunp77.medium.com/regression-algorithms-29f112797724. (Дата звернення: 04.12.2024).
13. Regression in Machine Learning: What It Is and Examples of Different Models. [Електронний ресурс], URL: https://builtin.com/data-science/regression-machine-learning. (Дата звернення: 04.12.2024).
14. The Drawbacks of Linear Regression Models: Limitations and Their Impact on Accuracy. [Електронний ресурс], URL: https://medium.com/@divine_inner_voice/the-drawbacks-of-linear-regression-models-limitations-and-their-impact-on-accuracy-675853072a3. (Дата звернення: 04.12.2024).
15. What is lasso regression? [Електронний ресурс], URL: https://www.ibm.com/topics/lasso-regression. (Дата звернення: 04.12.2024).
16. Ridge Regression. [Електронний ресурс], URL: https://medium.com/@abelkuriakose/ridge-regression-98c2a65cb3b1. (Дата звернення: 04.12.2024);
17. Understanding K-Nearest Neighbors (KNN) Regression in Machine Learning. [Електронний ресурс], URL: https://medium.com/@nandiniverma78988/understanding-k-nearest-neighbors-knn-regression-in-machine-learning-c751a7cf516c. (Дата звернення: 04.12.2024)
18. Unveiling Decision Tree Regression: Exploring its Principles, Implementation. [Електронний ресурс], URL: https://medium.com/@vk.viswa/unveiling-decision-tree-regression-exploring-its-principles-implementation-beb882d756c6. (Дата звернення: 04.12.2024).

19. Alakh Sethi. Support Vector Regression Tutorial for Machine Learning. [Електронний ресурс], URL: https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/. (Дата звернення: 04.12.2024).
20. Comparing Regression Models. [Електронний ресурс], URL: https://www.kaggle.com/code/ankitjha/comparing-regression-models. (Дата звернення: 04.12.2024).
21. Bishop, Christopher M. Pattern Recognition and Machine Learning. New York :Springer, 2006.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) ***Магістр***
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – Комп'ютерна інженерія
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
в.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«12» вересня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Румянцева Данила Максимовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«Модель комп'ютерної системи аналізу та прогнозування злочинності за допомогою методів машинного навчання»**

керівник роботи Бакуменко Ніна Станіславівна, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету ***№ 4101-5/3657 від 12 листопада 2024 року***

2. Строк подання студентом роботи ***30 листопада 2024 року***

3. Перелік питань, які потрібно розробити:

1. Постановка задачі аналізу та прогнозування даних.
2. Порівняльний аналіз методів аналізу та прогнозування даних.
3. Розробка математичної моделі аналізу та прогнозування злочинності за допомогою методів машинного навчання.
4. Розробка програмно-алгоритмічної реалізації моделі аналізу та прогнозування злочинності за допомогою методів машинного навчання.
5. Тестування та аналіз отриманих результатів

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Пошук та аналіз наукової літератури	05.09.2024 — 15.09.2024
2	Дослідження методів машинного навчання для моделювання залежностей між змінними	16.09.2024 — 01.10.2024
3	Обґрунтування відповідного методу побудови моделі	01.10.2024 — 15.10.2024
4	Розробка комп'ютерної реалізації моделі аналізу та прогнозування злочинності	16.10.2024 — 24.10.2024
5	Відлагодження та тестування розробленої комп'ютерної моделі	16.10.2024 — 17.11.2024
6	Корегування моделі після тестування	17.11.2024 — 20.11.2024
7	Оформлення пояснювальної записки	Листопад 2024
8	Передзахист кваліфікаційної роботи	Листопад 2024
9	Представлення кваліфікаційної роботи керівнику та рецензенту	28.11.2024 - 30.11.2024

5. Дата видачі завдання ***05 вересня 2024року.***

Студент Д. М. Румянцев _____
ініціали, прізвище підпис

Керівник роботи Н.С. Бакуменко _____
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«_____»_____2024 р.

Технічне завдання

на розробку програмного виробу «МОДЕЛЬ КОМП'ЮТЕРНОЇ СИСТЕМИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ЗЛОЧИННОСТІ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ»

1.	Введення	1.1. Назва: МОДЕЛЬ КОМП'ЮТЕРНОЇ СИСТЕМИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ЗЛОЧИННОСТІ ЗА ДОПОМОГОЮ МЕТОДІВ МАШИННОГО НАВЧАННЯ. 1.2. Галузь застосування: інформаційні технології.
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп’ютерна інженерія. 2.2. Завдання на кваліфікаційну роботу магістра ***№4101-5/3657*** від ***12 листопада 2024 року*** (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: дослідження методів машинного навчання для аналізу та прогнозування злочинності. 3.2. Призначення розробки: для аналізу та прогнозування злочинності. 3.3. Вихідні дані розробки: статистичні експериментальні дані.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик:немає. 4.2. Вимоги до надійності: забезепчувати точність та достовірність визначення інформативності параметрів стану не гіршу за існуючі методи та програмне забезепечення. 4.3.Вимоги до умов експлуатації: немає. 4.4. Вимоги до складу і параметрів технічних засобів: звичайне обчислювальне обладнання, ПК, тощо. 4.5. Вимоги до інформаційної та програмної сумісності: немає. 4.6. Вимоги до маркування та упаковки: немає. 4.7. Вимоги до транспортування і зберігання: на звичайних носіях інформації 4.8. Спеціальні вимоги: немає.

5.	Вимоги до програмної документації	Програмною документацією до виробу «Модель комп'ютерної системи аналізу та прогнозування злочинності за допомогою методів машинного навчання» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи). 3) Програму і методику випробувань розробленого програмного вироби.	
6.	Вимоги до техніко-економічних показників	Програмною документацією до виробу «Модель комп'ютерної системи аналізу та прогнозування злочинності за допомогою методів машинного навчання» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи).	
7.	Стадії і етапи розробки	Дата	Назва етапу
		від 12 червня 2024 до 29 червня 2024	Пошук та аналіз наукової літератури
		від 4 квітня 2024 до 15 травня 2024	Дослідження методів машинного навчання для моделювання залежностей між змінними
		від 8 травня 2024 до 10 вересня 2024	Обґрунтування відповідного методу побудови моделі
		від 10 вересня 2024 до 17 жовтня 2024	Розробка комп’ютерної реалізації моделі аналізу та прогнозування злочинності
		від 17 жовтня 2024 до 21 жовтня 2024	Відлагодження та тестування розробленої комп’ютерної моделі

		від 21 жовтня 2024 до 11 листопада 2024 від 12 листопада 2024 до 27 листопада 2024 від 27 листопада 2024 до 10 грудня 2024 20 грудня 2024	Корегування моделі після тестування Оформлення пояснювальної записки Передзахист кваліфікаційної роботи Представлення кваліфікаційної роботи керівнику та рецензенту
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику.	

Виконавець
студент групи КІ- 61
Румянцев Д. М._____

Замовник
к.т.н, доцент, доцент кафедри комп'ютерних систем та робототехніки
Бакуменко Н. С. _____

Додаток В

Програма та методика випробування програмного виробу «Модель комп'ютерної системи аналізу та прогнозування злочинності за допомогою методів машинного навчання»

1. Об’єкт випробувань

1. Назва програмного виробу: «Модель комп'ютерної системи аналізу та прогнозування злочинності за допомогою методів машинного навчання».
2. Область застосування: інформаційні технології.
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональних характеристик розробленої програми вимогам, зазначеним у Технічному завданні.

3. Загальні положення

1. Підстави щодо випробувань

Підставою до випробувань є: 1) Навчальний план за спеціальністю 123 – «Комп’ютерна інженерія»; 2) Завдання на кваліфікаційну роботу затверджено наказом № 4101-5/*3657*; 3) Наказ про призначення атестаційної комісії.

2. Місце та тривалість випробувань

Приймальні випробування проводяться на базі комп’ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного виробу проводяться в обсязі відповідно до цієї програми та методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програмного виробу

Модель повинна задовольняти наступним вимогам:

1. Програма повинна являти собою комп’ютерну реалізацію моделі аналізу та прогнозування злочинності за допомогою методів машинного навчання;
2. безпомилково виконуватися на будь-якому комп’ютері, який задовольняє вимогам, при правильних початкових даних;
3. забезпечувати можливість виведення результату на екран;
4. програма повинна бути стійкою до помилок;
5. для виконання програми необхідний ПК з ОС «Windows» або «Linux», а також доступом до Інтернету для Google Colab;
6. програма призначена для комп’ютерів з мінімальними характеристиками: процесор з тактовою частотою не нижче 1 ГГц; оперативний запам’ятовуючий пристрій – 1024 Мб;
7. вимоги до складу і параметрів технічних засобів;
8. вимоги до маркування та упаковки (не висуваються);
9. вимоги до транспортування і зберігання (не висуваються).

Спеціальні вимоги (не висуваються).

5. Вимоги до програмної документації

Програмною документацією до виробу «Модель комп'ютерної системи аналізу та прогнозування злочинності за допомогою методів машинного навчання» вважати:

1. справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);
2. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);
3. рекомендацій щодо застосування створеної програмної стандартизації у проєктах (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).
4. Текст програми (представити як Додаток Г до пояснювальної записки до кваліфікаційної роботи).

6. Засоби та порядок випробувань

6.1 Засоби випробувань

Випробування проводяться на ПК з ОС «Windows» у Google Colab.

6.2 Порядок проведення випробувань

1-й етап.

- Перевірка комплектності програмної документації

Перевірка комплектності представленої програмної документації здійснюється за критерієм відповідності її складу

Технічному завданню. Перевірка комплектності технічних та програмних засобів здійснюється за критерієм відповідності їх складу Технічному завданню.

- Перевірка якості представленої на випробування програмної документації на відповідність вимогам стандартів ЄСПД.

Якість програмної документації відповідає вимогам ДЕСТ 19301-79 ЄСПД.

2-й етап

Перевірка працездатності розробленої програми.

Тест 1: Перевірка роботи побудови регресійних моделей

Тест вважається пройденим, якщо при компіляції програми не виникає будь-яких помилок (у консольному вікні відсутний текст червоного кольору і справа виводиться ділянка завантаження).

Запуск програми здійснюється натисканням ctrl + enter після вибору комірки коду. Результат виконання зображено на рисунку 1.

```
[77] lasso_regression = Lasso(max_iter=10000).fit(xtrain, ytrain)
     print("Training set score:", round(lasso_regression.score(xtrain, ytrain), 3))
     print("Testing set score:", round(lasso_regression.score(xtest,ytest),3))
     print("Coefficients :", lasso_regression.coef_)
     print("Intercept:", lasso_regression.intercept_)

===Lasso Regression (default parameters):===
Training set score: 0.958
Testing set score: 0.923
Coefficients : [ 7.01660361 -0.11801716  0.         -0.          0.          0.
  0.         -0.          0.         48.61842725 -0.50611236 -0.
 -0.          0.         -0.          0.         -0.          0.        ]
Intercept: [7.72192759]

[78] print("\n===Knn regression (default parameters):===")
     knn_model = KNeighborsRegressor().fit(xtrain, ytrain)
     print("Training set score:")
     print(round(knn_model.score(xtrain, ytrain), 3))
     print("Testing set score:")
     print(round(knn_model.score(xtest, ytest), 3))

===Knn regression (default parameters):===
Training set score:
0.705
Testing set score:
0.74
```

Рис. В.1 – Результат виконання Тесту 1

В результаті виконання Тесту 1 програма вважається працездатною.

Тест 2: Перевірка роботи крос-валідації

Тест вважається пройденим, якщо при компіляції програми не виникає будь-яких помилок (у консольному вікні відсутний текст червоного кольору і справа виводиться ділянка завантаження).

Запуск програми здійснюється натисканням ctrl + enter після вибору комірки коду. Результат виконання зображено на рисунку 2.

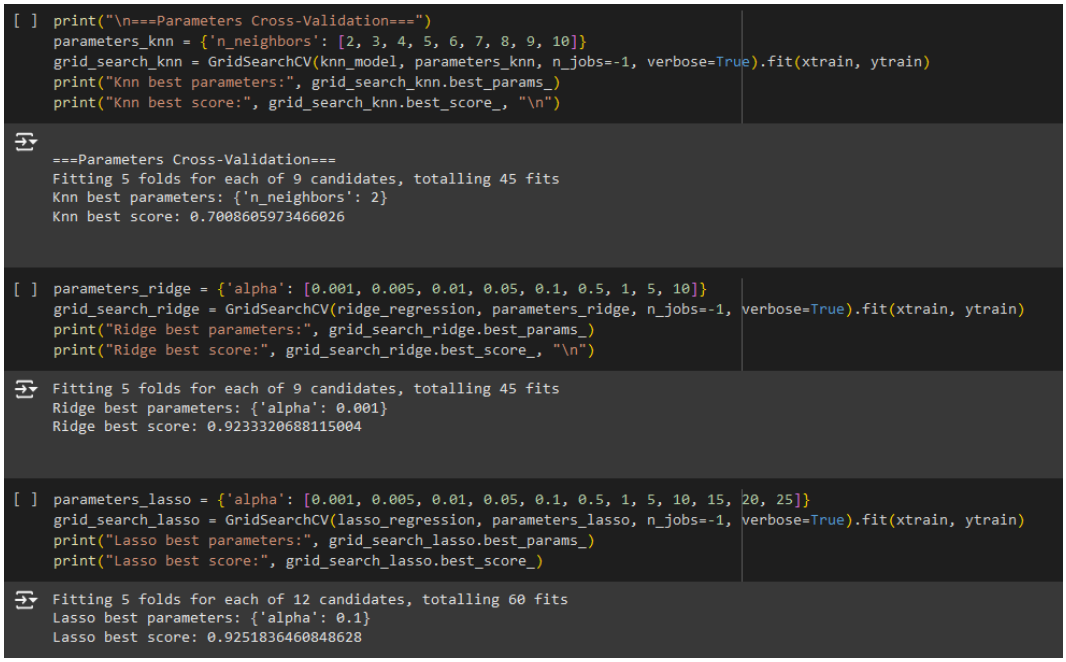


Рис. В.2 – Результат виконання Тесту 2

В результаті виконання Тесту 2 програма вважається працездатною.

Тест 3: Перевірка роботи нейронної мережі

Тест вважається пройденим, якщо при компіляції програми не виникає будь-яких помилок (у консольному вікні відсутний текст червоного кольору і справа виводиться ділянка завантаження).

Запуск програми здійснюється натисканням ctrl + enter після вибору комірки коду. Результат виконання зображено на рисунку 3.

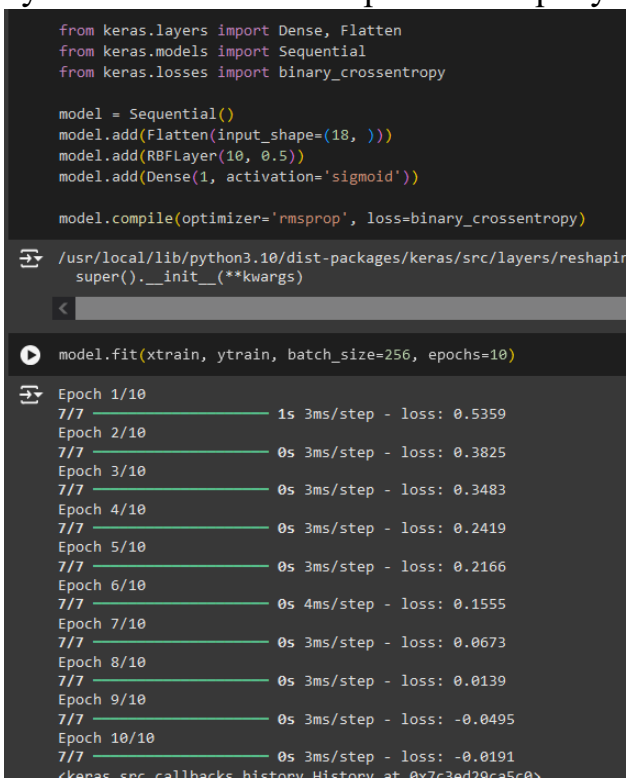


Рис. В.3 – Результат виконання Тесту 3

В результаті виконання Тесту 3 програма вважається працездатною.

Висновки: усі 3 тести успішно пройшли випробування. Отже випробування пройшло успішно.

Виконавець: студент групи КІ-61, Румянцев Д. М. 

Додаток Г

Лістинг коду програми

```
      import numpy as np
      import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt
      from sklearn.linear_model import LinearRegression, Ridge, Lasso
      from sklearn.model_selection import train_test_split,
GridSearchCV
      from sklearn.neighbors import KNeighborsRegressor
      from sklearn.preprocessing import StandardScaler
      from sklearn.tree import DecisionTreeRegressor

      uscrime = pd.read_csv("/content/drive/My Drive/crimedata.txt")
      uscrime_df = pd.DataFrame(uscrime)
      uscrime_df = uscrime_df.replace('?', np.nan)

      print("----------------------------- US Crime Data --------------
-------------")
      print(uscrime_df.head(10))
      print(uscrime_df.describe())
      print("NaN values:")
      print(uscrime_df.isnull().sum())
      print("Total amount of NaN values: %s" %
uscrime_df.isnull().sum().sum())

      #Removing non-predictive features
      uscrime_df =
uscrime_df.drop(columns=['state','countyCode','communityCode','communi
tyname','fold'], axis=1)
      #Filling all NaN with median values
      feat_miss = uscrime_df.columns[uscrime_df.isnull().any()]
      uscrime_df[uscrime_df.columns] =
uscrime_df[uscrime_df.columns].apply(pd.to_numeric, errors='coerce')
      uscrime_df.fillna(uscrime_df[feat_miss].median(),inplace=True)



      #The new dataset
      print("\n---------------------- US Crime Data (formated) --------
---------------")
      print(uscrime_df.head(10))
      print(uscrime_df.describe())
      print("NaN values:")
      print(uscrime_df.isnull().sum())
      print("Total amount of NaN values: %s" %
uscrime_df.isnull().sum().sum())

      X_train1 = uscrime_df[['population','householdsize',
'agePct12t21','agePct12t29','agePct16t24','agePct65up',

'murders','murdPerPop','rapes','rapesPerPop','robberies','robbbPerPop'
,
```

Продовження лістінгу

```
     'assaults','assaultPerPop','burglaries','burglPerPop','larcenies'
,'larcPerPop',

'autoTheft','autoTheftPerPop','arsons','arsonsPerPop','ViolentCrimesPe
rPop','nonViolPerPop']]
     X_train2 = uscrime_df[['numbUrban','pctUrban',
'medIncome','pctWWage','pctWFarmSelf','pctWInvInc', 'pctWSocSec',

'pctWPubAsst','pctWRetire','medFamInc','perCapInc',

'murders','murdPerPop','rapes','rapesPerPop','robberies','robbbPerPop'
,

'assaults','assaultPerPop','burglaries','burglPerPop','larcenies','lar
cPerPop',

'autoTheft','autoTheftPerPop','arsons','arsonsPerPop','ViolentCrimesPe
rPop','nonViolPerPop']]
     X_train3 =
uscrime_df[['NumUnderPov','PctPopUnderPov','PctLess9thGrade','PctNotHS
Grad','PctBSorMore','PctUnemployed',

'PctEmploy','PctEmplManu','PctEmplProfServ','PctOccupManu','PctOccupMg
mtProf',

'murders','murdPerPop','rapes','rapesPerPop','robberies','robbbPerPop'
,

'assaults','assaultPerPop','burglaries','burglPerPop','larcenies','lar
cPerPop',

'autoTheft','autoTheftPerPop','arsons','arsonsPerPop','ViolentCrimesPe
rPop','nonViolPerPop']]

     for x in [X_train1, X_train2, X_train3]:
         corr_matrix = x.corr(method='pearson')
         plt.figure()
         sns.heatmap(corr_matrix, annot=True, annot_kws={'size': 7})
         plt.show()

     X_train =
uscrime_df[['population','agePct12t29','numbUrban','pctUrban','medInco
me','pctWInvInc',

'pctWSocSec','pctWPubAsst','medFamInc','NumUnderPov','PctPopUnderPov',

'PctLess9thGrade','PctNotHSGrad','PctBSorMore','PctUnemployed',

'PctEmploy','PctOccupManu','PctOccupMgmtProf']]

     target = uscrime_df[['murders']]

     std_scaler = StandardScaler()
     scaled_X_train = std_scaler.fit_transform(X_train)
```

Продовження лістінгу

```
      X_train_scaled = pd.DataFrame(scaled_X_train)

      xtrain, xtest, ytrain, ytest = train_test_split(X_train_scaled,
target, test_size=0.3, random_state=5)

      print("\n===Linear Regression (default parameters):===")
      linear_regression = LinearRegression().fit(xtrain, ytrain)
      print("Training set score:",
round(linear_regression.score(xtrain, ytrain), 3))
      print("Testing set score:",
round(linear_regression.score(xtest,ytest),3))
      print("Coefficients :", linear_regression.coef_)
      print("Intercept:", linear_regression.intercept_)

      print("\n===Ridge Regression (default parameters):===")
      ridge_regression = Ridge().fit(xtrain, ytrain)
      print("Training set score:", round(ridge_regression.score(xtrain,
ytrain), 3))
      print("Testing set score:",
round(ridge_regression.score(xtest,ytest),3))
      print("Coefficients :", ridge_regression.coef_)
      print("Intercept:", ridge_regression.intercept_)

      print("\n===Lasso Regression (default parameters):===")
      lasso_regression = Lasso(max_iter=10000).fit(xtrain, ytrain)
      print("Training set score:", round(lasso_regression.score(xtrain,
ytrain), 3))
      print("Testing set score:",
round(lasso_regression.score(xtest,ytest),3))
      print("Coefficients :", lasso_regression.coef_)
      print("Intercept:", lasso_regression.intercept_)

      print("\n===Knn regression (default parameters):===")
      knn_model = KNeighborsRegressor().fit(xtrain, ytrain)
      print("Training set score:")
      print(round(knn_model.score(xtrain, ytrain), 3))
      print("Testing set score:")
      print(round(knn_model.score(xtest, ytest), 3))

      print("\n===Decision Tree Regression (default parameters):===")
      decision_tree_regression = DecisionTreeRegressor().fit(xtrain,
ytrain)
      print("Training set score:",
round(decision_tree_regression.score(xtrain, ytrain), 3))
      print("Testing set score:",
round(decision_tree_regression.score(xtest, ytest), 3))

      print("\n===Parameters Cross-Validation===")
      parameters_knn = {'n_neighbors': [2, 3, 4, 5, 6, 7, 8, 9, 10]}
      grid_search_knn = GridSearchCV(knn_model, parameters_knn,
n_jobs=-1, verbose=True).fit(xtrain, ytrain)
      print("Knn best parameters:", grid_search_knn.best_params_)
      print("Knn best score:", grid_search_knn.best_score_, "\n")
```

Продовження лістінгу

```
    parameters_ridge = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5,
1, 5, 10]}

    grid_search_ridge = GridSearchCV(ridge_regression,
parameters_ridge, n_jobs=-1, verbose=True).fit(xtrain, ytrain)
    print("Ridge best parameters:", grid_search_ridge.best_params_)
    print("Ridge best score:", grid_search_ridge.best_score_, "\n")

    parameters_lasso = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5,
1, 5, 10, 15, 20, 25]}
    grid_search_lasso = GridSearchCV(lasso_regression,
parameters_lasso, n_jobs=-1, verbose=True).fit(xtrain, ytrain)
    print("Lasso best parameters:", grid_search_lasso.best_params_)
    print("Lasso best score:", grid_search_lasso.best_score_)

    print("\n===Ridge Regression (best parameters):===")
    ridge_regression_best = Ridge(alpha=0.001).fit(xtrain, ytrain)
    print("Training set score:",
round(ridge_regression_best.score(xtrain, ytrain), 3))
    print("Testing set score:",
round(ridge_regression_best.score(xtest, ytest), 3))
    print("Coefficients :", ridge_regression_best.coef_)
    print("Intercept:", ridge_regression_best.intercept_)

    print("\n===Lasso Regression (best parameters):===")
    lasso_regression_best = Lasso(alpha=0.1,
max_iter=10000).fit(xtrain, ytrain)
    print("Training set score:",
round(lasso_regression_best.score(xtrain, ytrain), 3))
    print("Testing set score:",
round(lasso_regression_best.score(xtest, ytest), 3))
    print("Coefficients :", lasso_regression_best.coef_)
    print("Intercept:", lasso_regression_best.intercept_)

    print("\n===Knn regression (best parameters):===")
    knn_model = KNeighborsRegressor(n_neighbors=2).fit(xtrain,
ytrain)
    print("Training set score:")
    print(round(knn_model.score(xtrain, ytrain), 3))
    print("Testing set score:")
    print(round(knn_model.score(xtest, ytest), 3))

    from keras.layers import Layer
    import tensorflow.keras.backend as K
    import tensorflow as tf


    class RBFLayer(Layer):
        def __init__(self, units, gamma, **kwargs):
            super(RBFLayer, self).__init__(**kwargs)
            self.units = units
            self.gamma = tf.cast(gamma, dtype=K.floatx())
```

Продовження лістінгу

```
    def build(self, input_shape):
            self.mu = self.add_weight(name='mu',
                                      shape=(int(input_shape[1]),
self.units),
                                      initializer='uniform',
                                      trainable=True)

            super(RBFLayer, self).build(input_shape)

        def call(self, inputs):
            diff = tf.expand_dims(inputs,-1) - self.mu
            l2 = K.sum(K.pow(diff, 2), axis=1)
            res = K.exp(-1 * self.gamma * l2)
            return res

        def compute_output_shape(self, input_shape):
            return (input_shape[0], self.units)


    from keras.layers import Dense, Flatten
    from keras.models import Sequential
    from keras.losses import binary_crossentropy

    model = Sequential()
    model.add(Flatten(input_shape=(18, )))
    model.add(RBFLayer(18, 0.5))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer='rmsprop', loss=binary_crossentropy)

    model.fit(xtrain, ytrain, batch_size=256, epochs=8)

    test_loss = model.evaluate(xtest, ytest)
    print("test loss", test_loss)
```