**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет**
**імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

**ДОПУЩЕНО ДО ЗАХИСТУ**
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____202_ р.

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

# ІНТЕЛЕКТУАЛЬНА СИСТЕМА ДЛЯ ПРОГНОЗУВАННЯ ВИХІДНОЇ ЕЛЕКТРОЕНЕРГІЇ ЕЛЕКТРОСТАНЦІЇ КОМБІНОВАНОГО ТИПУ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

Спеціальність 122 «Комп'ютерні науки»

**122 – МКР – 601.21610404**

**Студент** _____ М.В. Борисов
«___» _____ 2022 р.

**Консультант** _____ О. П. Гожий
доктор технічних наук, професор
«___» _____ 2022 р.

**Миколаїв – 2022**

**Чорноморський національний університет ім. Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інтелектуальних інформаційних систем**

Освітньо-кваліфікаційний рівень **магістр**

Галузь знань **12 «Інформаційні технології»**

*(шифр і назва)*

Спеціальність **122 «Комп'ютерні науки»**

*(шифр і назва)*

**ЗАТВЕРДЖУЮ**

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.

_____ Ю. П. Кондратенко

«____» _____20__р.

# ЗАВДАННЯ

## на магістерську кваліфікаційну роботу

**БОРИСОВУ Миколі Віталійовичу**

1. Тема магістерської кваліфікаційної роботи «Інтелектуальна система для прогнозування вихідної електроенергії електростанції комбінованого типу на основі методів машинного навчання».

**Керівник роботи Гожий Олександр Петрович д-р техн. наук, професор.**

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____20___р. № _____

2. Строк подання студентом роботи «___» ___ 20___р.

3. Вхідні (початкові) дані до роботи: для виконання роботи не потрібні початкові дані.

Очікуваний результат роботи: Інтелектуальна система прогнозування вихідної електроенергії електростанції комбінованого типу.

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):

– аналіз сучасного стану електростанцій;

– огляд основних методів регресії для прогнозування;

– розробка інтелектуальної системи для прогнозування вихідної електроенергії електростанції комбінованого типу на основі методів машинного навчання.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини:

_____

_____

_____.

7. Консультанти:

| **Розділ** | **Прізвище, ініціали та посада консультанта** | **Підпис** |
|---|---|---|
| Спеціальна частина з охорони праці | | |
| Методична частина | | |

Керівник роботи д.т.н, професор, Гожий О. П.

_____

*(підпис)*

Завдання прийнято до виконання Борисов М.В.

*(прізвище та ініціали)*

_____

*(підпис)*

Дата видачі завдання «____» _____ 20__ р.

**КАЛЕНДАРНИЙ ПЛАН**

**Виконання магістерської кваліфікаційної роботи**

Тема: Інтелектуальна система для прогнозування вихідної електроенергії електростанції комбінованого типу на основі методів машинного навчання.

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1 | Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР | 05.09.2021 | 09.10.2021 | |
| 2 | Отримання завдання на виконання МКР | 20.10.2021 | 23.10.2021 | |
| 3 | Складання календарного плану на період виконання МКР | 24.10.2021 | 28.10.2021 | |
| 4 | Огляд літератури за темою дослідження | 29.10.2021 | 13.11.2021 | |
| 5 | Проходження переддипломної практики, збір та аналіз матеріалів до МКР | 22.11.2021 | 11.12.2021 | |
| 6 | Аналіз предметної області та розробка технічного завдання. Моделювання результатів | 14.12.2021 | 11.01.2022 | |
| 7 | Опис фахової частини МКР | 12.01.2022 | 24.01.2022 | |
| 8 | Розробка спеціальної частини з охорони праці та методичної частини | 25.01.2022 | 30.01.2022 | |
| 9 | Попередній захист МКР на засіданні комісії кафедри | 31.01.2022 | 31.01.2022 | |
| 10 | Корегування роботи за результатами попереднього захисту | 01.02.2022 | 03.02.2022 | |
| 11 | Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту | 04.02.2022 | 06.02.2022 | |
| 12 | Подання МКР рецензенту | 09.02.2022 | 10.02.2022 | |
| 13 | Рецензування МКР | 11.02.2022 | 12.02.2022 | |
| 14 | Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 14.02.2022 | 15.02.2022 | |
| 15 | Захист МКР перед екзаменаційною комісією (ЕК) | 21.02.2022 | 22.02.2022 | |

Розробив студент Борисов М.В. _____

(*підпис*)

Керівник роботи д.т.н, професор, Гожий О. П. _____

*(підпис)*

«____» _____ 2022 р.

# АНОТАЦІЯ

до магістерської кваліфікаційної роботи
студента групи 601 ЧНУ ім. Петра Могили

**Борисова Миколи Віталійовича**

на тему: **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА ДЛЯ ПРОГНОЗУВАННЯ ВИХІДНОЇ ЕЛЕКТРОЕНЕРГІЇ ЕЛЕКТРОСТАНЦІЇ КОМБІНОВАНОГО ТИПУ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ“**


**Актуальність** даного дослідження полягає у необхідності підвищення ефективності електростанцій комбінованого типу, розробці інтелектуальної системи з використанням різних методів регресії для вирішення відповідної задачі. Це дозволить збільшити ККД електростанцій та зменшить витрати на пальне.

**Об’єктом** дослідження є технології створення інтелектуальної системи передбачення на основі методів машинного навчання.

**Предметом** дослідження є різні методи регресії, які використовуються для прогнозування вихідної електроенергії.

**Метою** дослідження є створення найбільш якісної моделі прогнозування для її подальшого використані на електростанції комбінованого типу заради підвищення ефективності роботи.

В результаті виконання роботи було створено та досліджено шість регресійних методів для прогнозування(проста лінійна регресія, множинна регресія, дерево рішень, та три ансамблеві методи), проаналізовано вплив усіх незалежних змінних на якість прогнозування, а також застосовано основні метрики для оцінки якості прогнозування та вибору ліпшої моделі.

Дана робота складається з п’яти розділів. Кожен розділ відповідно присвячений: аналізу методів машинного навчання, технології та підходи до проектування системи, програмна реалізація системи для прогнозування вихідної електроенергії електростанції комбінованого типу, спеціальної методичної частини та спеціальної частини з охороні праці та безпеки у надзвичайних ситуаціях. Загальний обсяг роботи – 94 сторінки. Магістерська кваліфікаційна робота містить один додаток, 59 рисунків, 7 таблиць і посилання на 40 літературних джерел.



**Ключові слова**: машинне навчання, методи регресії, предиктор, мітка, метрики оцінки якості прогнозування.

# ABSTRACT

to the master's qualification work by the student of the group 601
of Petro Mohyla Black Sea National University

**Borysov Mykola**

## “ INTELLECTUAL SYSTEM FOR FORECASTING OUTPUT ELECTRICITY OF COMBINED TYPE POWER PLANT BASED ON MACHINE TRAINING METHODS ”

**The relevance** of this study is the need to increase the efficiency of combined power plants, the development of an intelligent system using different regression methods to solve the problem. This will increase the efficiency of power plants and reduce fuel costs.

**The object** of research is the technology of creating an intelligent prediction system based on machine learning methods.

**The subject** of the study are various regression methods used to predict the output electricity.

**The aim** of the study is to create the highest quality forecasting model for its further use in combined power plants to improve efficiency.

As a result of the work, six regression methods for forecasting (simple linear regression, multiple regression, decision tree, and three ensemble methods) were developed and investigated, the impact of all independent variables on forecasting quality was analyzed, and basic metrics were used to assess forecasting and selection quality. better model.

This work consists of five sections. Each section is devoted to: analysis of machine learning methods, technologies and approaches to system design, software implementation of the system for forecasting power output of the combined type power plant, special methodological part and special part on labor protection and safety in emergencies. Total workload – 94 pages. The master's thesis contains one appendix, 59 figures, 7 tables and references to 40 references.

**Key words:** machine learning, regression methods, predictor, label, forecasting quality assessment metrics.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

МН – машинне навчання.

МО- метрики оцінювання.

ІС – інтелектуальна система.

СР – середовище розробки.

МР – магістерська робота.

РМ – регресійні методи.

# Пояснювальна записка

**до магістерської кваліфікаційної роботи**

на тему:

**«ІНТЕЛЕКТУАЛЬНА СИСТЕМА ДЛЯ ПРОГНОЗУВАННЯ ВИХІДНОЇ ЕЛЕКТРОЕНЕРГІЇ ЕЛЕКТРОСТАНЦІЇ КОМБІНОВАНОГО ТИПУ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ»**

Спеціальність 122 «Комп'ютерні науки та інформаційні технології»

**122 – МКР – 601.21610404**

**Виконав:** студент 6 курсу, групи 601
М. В. Борисов
(підпис, ініціали та прізвище)
«___» _____ 2022 р

**Керівник МКР:** д.т.н., професор, професор
кафедри ІІТ ЧНУ ім. Петра Могили
О.П. Гожий
(підпис, ініціали та прізвище)
«___» _____ 2022 р

**Миколаїв – 2022**

# ВСТУП

*Згідно із оцінкою McKinseyGlobalInstitute, у 2018 році лише у США попит на експертів із машинного навчання буде перевищувати пропозицію на 140-190 т. чоловік. Крім того, буде необхідно півтора мільйони управлінців, що розбираються у даних.*

*(Домігос П. Верховний алгоритм: як машинне навчання змінить наш світ. М. : Манн, Іванов та Фербер, 2016.)*

На сьогоднішній день одною з найбільш актуальних проблем людства є проблема добування електроенергії, найпоширеніший спосіб якого, є використання природних джерел пального, що тягне за собою дві інші величезні проблеми, це забруднення навколишнього середовища, та дефіцит природних копалин таких, як газ, нафта та вугілля. Оскільки людство може опинитися на порозі світової екологічної катастрофи, вже сьогодні необхідно приймати рішення щодо переходу до використання екологічно чистих та відновлювальних джерел енергії таких як гідроелектростанції, вітрові електростанції та інші.

Не всі країни можуть відмовитись від класичних видів добування електроенергії, деякі не мають виходу до водойм, тому не можуть встановити гідроелектростанції. Деяким країнам клімат не дозволяє встановлювати сонячні електростанції. В Україні ця проблема навіть більш актуальна, тому що через непросту зовнішню політику, наша країна має проблеми у сфері енергетики. Близько 60% від усіх електростанцій у нашій країні становлять ТЕС, 12,4% - гідроелектростанції, а частка вітрових та сонячних електростанцій становить лише 0,5% від усієї електроенергії. Тому необхідно розглядати інші варіанти добування електроенергії.

Одним з таким варіантів є електростанція комбінованого циклу. В основі даного виробництва лежить класична газотурбінна установка, тобто при згорянні газової суміші пара крутить лопаті турбіни, яка виробляє електроенергію. ККД даної станції приблизно 33%, інші 67% це відходи.

Інше рішення це газокомбінованний цикл в якому, пропонується додавати до газової турбіни парову. Після відпрацювання у газовій турбіні вихлопні гази залишаються ще досі гарячими (140˚С), тому ці відходи можна використовувати для нагрівання води для парової турбіни, що підвищує ККД до 68%.

Актуальність теми роботи полягає у тому, що створена інтелектуальна система допоможе визначити які чинники впливають на потужність електростанції, та спрогнозує вихідну електроенергію, в залежності від різних значень цих чинників.

**Об’єктом дослідження** є технології створення інтелектуальної системи передбачення на основі методів машинного навчання.

**Предметом дослідження** є різні методи регресії, які використовуються для прогнозування вихідної електроенергії.

# 1АНАЛІЗ МЕТОДІВ МАШИННОГО НАВЧАННЯ. ПОСТАНОВКА ЗАДАЧІ

Методи машинного навчання розповсюджуються в сучасному світі та знаходять використання у все більшій кількості різних сфер життя, таких як:

1) компютерне бачення;
2) розпізнавання мови;
3) компютерна лінгвістика;
4) медична діагностика;
5) біоінформатика;
6) технічна діагностика;
7) інформаційний пошук;
8) фінансові застосунки.

Тому їх використання є дуже актуальним та затребуваним. Раніше такі методи були складними та майже неможливими завданнями для комп’ютерів. Але зі збільшенням обчислювальної потужності та використанням методів машинного навчання з'явилися нові методи вирішення багатьох проблем, які раніше було складно або неможливо розв'язати.

Оскільки проблеми бувають різного характеру, то і методів вирішення існує багато. Тому навіть сьогодні знаходять багато нових методів та вдосконалюють старі. Складність полягає в величезній кількості параметрів під час налаштуванні під конкретну задачу. Але висока еластичність та швидкість роботи роблять методи машинного навчання дуже зручним інструментом.

Використовуючи відкриті джерела можна отримати датасети для створення суспільно-корисних та актуальних тем. Саме відкриті джерела дозволяють людям досліджувати реальні проблеми та запропонувати різні підходи для вирішення різних проблем. Прикладом такого набору даних є дослідження роботи електростанції комбінованого циклу протягом 2006-2011 років.

Електростанція комбінованого циклу – це електростанція, яка крім звичайної газової турбіни має ще й парову турбіну, які працюють разом (Рис. 1.1).

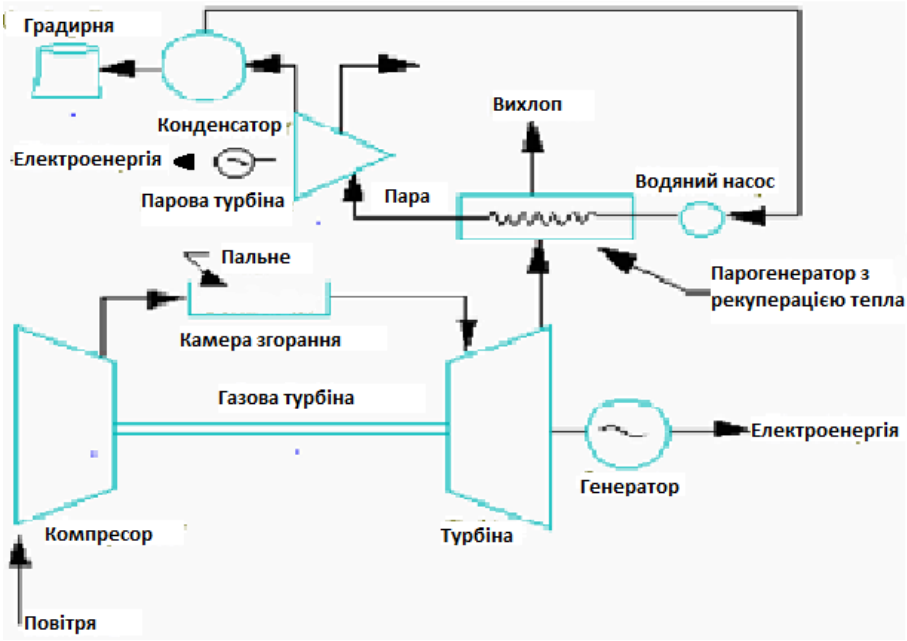



Рис. 1.1. Схема роботи газотурбінного циклу

## 1.1 Опис предметної сфери

### 1.1.1 Задачі машинного навчання

**Машинне навчання** — це підгалузь штучного інтелекту в галузі інформатики, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися» (тобто, поступово покращувати продуктивність у певній задачі) з даних, без того, щоби бути програмованими явно.Наприклад, система машинного навчання може бути натренована на електронних повідомленнях для розрізняння спаму і прийнятних повідомлень. Після навчання вона може бути використана для класифікації нових повідомлень електронної пошти на спам та не-спам. В основі машинного навчання розглядаються уявлення та узагальнення. Представлення даних і функцій оцінки цих даних є частиною всіх систем машинного навчання, наприклад, у наведеному вище прикладі, повідомлення електронною поштою, можна уявити лист як набір англійських слів, просто відмовившись від порядку слів. Узагальнення є властивістю, яку система буде застосовувати добре на

невидимих примірниках даних.Умови, за яких це може бути гарантовано, є ключовим об'єктом вивчення в полі обчислювальної теорії навчання. Існує широкий спектр завдань машинного навчання та успішних застосувань. Оптичне розпізнавання символів, в яких друковані символи розпізнаються автоматично, та ґрунтуються на попередніх прикладах, є класичним підходом техніки машинного навчання. 1959 року Артур Самуїл визначив машинне навчання як «Поле дослідження, яке дає комп'ютерам можливість навчатися, не будучи явно запрограмованими».

Задачі машинного навчання, як правило, поділяють на дві широкі категорії, залежно від того, чи доступний системі, що навчається, навчальний «сигнал», або «зворотний зв'язок»:

1) **навчання з учителем**: комп'ютерові представляють приклади входів та їхніх бажаних виходів, задані «вчителем», і метою є навчання загального правила, яке відображає входи на виходи. В окремих випадках вхідний сигнал може бути доступним лише частково, або бути обмеженим особливим зворотним зв'язком;
2) **навчання без учителя**: алгоритмові навчання не дається міток, залишаючи його самому знаходити структуру в своєму вході. Навчання без учителя може бути метою саме по собі (виявлення прихованих закономірностей у даних), або засобом досягнення мети (навчання ознак).
3) **навчання з підкріпленням**: метод машинного навчання, у якому відбувається навчання моделі, яка має відомостей про систему, але має можливість робити будь-які дії у ній. Події переводять систему в новий стан і модель отримує від системи певну винагороду.

Інша класифікація завдань машинного навчання виникає при розгляді бажаного *виходу* системи з машинним навчанням:

1) у класифікації входи поділяються на два або більше класів, і система-учень мусить породити модель, яка відносить небачені входи до одного або більше (багатоміткова класифікація) з цих класів. Це, як правило,

намагаються розв'язувати керованим чином. Прикладом класифікації є фільтри спаму, в яких входами є повідомлення електронної пошти (або чогось іншого), а класами є «спам» та «не спам»;

2) у регресії , також керованій задачі, виходи є безперервними, а не дискретними;
3) у кластеруванні набір входів повинен бути поділенний на групи. На відміну від класифікації, групи не відомі заздалегідь, що зазвичай робить це завданням для спонтанного навчання;
4) оцінка густини знаходить розподіл входів у деякому просторі;
5) зниження розмірності спрощує входи шляхом відображення їх на простір меншої розмірності. Пов'язаною задачею є тематичне моделювання, в якому програмі надають перелік документів людською мовою, і ставлять задачу з'ясувати, які документи охоплюють подібні теми.

Штучний інтелект – це розділ комп'ютерної лінгвістики та інформатики, що опікується формалізацією проблем та завдань, які подібні до дій, що виконує людина. Іншими словами, штучний інтелект – це здатність інженерної системи обробляти, застосовувати та вдосконалювати здобуті знання та вміння.

У більшості випадків алгоритм розв'язання завдання невідомий наперед. Точного визначення цієї науки немає, оскільки у філософії не розв'язано питання про природу і статус людського інтелекту. Немає і точного критерію досягнення комп'ютером «*розумності*», хоча перед штучним інтелектом було запропоновано низку гіпотез, наприклад, тест Тюрінга або гіпотеза Ньюелла-Саймона. Нині існує багато підходів як до розуміння задач штучного інтелекту, так і до створення інтелектуальних систем.

Глибинне навчання – це галузь машинного навчання (Рис. 1.2), що ґрунтується на наборі алгоритмів, які намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень.


Рис. 1.2. Розділи штучного інтелекту

Найчастіше глибинне навчання використовують у машинному зорі. Вирізняють декілька задач, які вирішують глибинним навчанням:

1) класифікація зображень. Класифікація зображень передбачає визначення класів об'єктів на зображеннях, наприклад, для відокремлення дефектних компонентів від бездефектних та їх сортування за різними типами дефектів або розподілу відпускних зображень за різними категоріями. Так, наприклад, у виробництві печива це завдання полягатиме у перевірці кожного печива на предмет того, чи не розкришилося воно;

2) сегментація та розпізнавання зображень. Сегментація зображень передбачає визначення класу кожного з пікселів зображення. Такий підхід дозволяє ідентифікувати декілька різних об'єктів на зображенні, наприклад, визначати різні види фруктів у кошику або розпізнавати вказівники, дорожні знаки та людей у потоці;

3) обробка зображень. Глибоке навчання також використовується на етапі обробки та оптимізації зображень, наприклад для видалення шуму або компенсації спотворень, характерних для різних об'єктивів.

### 1.1.2 Методи регресії

**Лінійна регресія** - це широко використовувана модель, що використовується для оцінки реальних величин і відповідає критерію безперервних змінних. У лінійній регресії взаємозв'язок між незалежними змінними та залежними змінними дотримується через лінію, яка зазвичай представляє зв'язок між двома змінними[2].

Лінія підгонки відома як лінія регресії і представлена лінійним рівнянням типу:

$$Y = a \cdot X + b \tag{1.1}$$

Формула заснована на інтерполяції даних, щоб пов'язувати дві або більше характеристик між собою. Коли алгоритму дається вхідна характеристика, регресія повертає іншу характеристику.

Коли є більше однієї незалежної змінної, тоді слід використовувати множинні лінійні регресії, припускаючи модель на зразок наступної:

$$Y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_n \cdot x_n, \tag{1.2}$$

де y - відповідь на величини, тобто представляє результат, передбачений моделлю;

$b_0$- перехоплення , тобто значення y, коли $x_i$ всі вони рівні 0;

$b_1$- коефіцієнт $x_1$;

$b_n$–коефіцієнт $x_n$;

$x_1, x_2, \ldots, x_n$ – є незалежними змінними моделі.

В основному рівняння пояснює взаємозв'язок між суцільною залежною змінною (y) та двома або більше незалежними змінними (x1, x2, x3…).

Множинна лінійна регресія виявляє більший інтерес у галузі машинного навчання та штучного інтелекту, оскільки дозволяє отримати виконуючі моделі навчання навіть у випадку великої кількості записів, що підлягають аналізу.

**Логістична регресія** - це статистичний інструмент, спрямований на моделювання біноміального результату з однією або кількома пояснювальними змінними.

Зазвичай використовується для бінарних проблем, коли є лише два класи, наприклад, так чи ні, 0 або 1, чоловіки чи жінки тощо.

Таким чином можна описати дані та пояснити зв’язок між бінарною залежною змінною та однією або кількома номінальними чи порядковими незалежними змінними.

Результат визначається завдяки використанню логістичної функції, яка оцінює ймовірність, а потім визначає найближчий клас (позитивний чи негативний) до отриманого значення ймовірності.

**Дерево рішень** - це деревоподібний графік (Рис 1.3), де сортування починається від кореневого вузла до вузла листя до досягнення мети. Він є найпопулярнішим для прийняття рішень та класифікації на основі керованих алгоритмів. Він побудований за допомогою рекурсивного розподілу, де кожен вузол виступає як тестовий випадок для деяких атрибутів, а кожен край, що походить від вузла, є можливою відповіддю в тестовому випадку. І кореневий, і лістовий вузли - два об'єкти алгоритму.

У алгоритмі регресії рішення або результат є безперервним. Він отримує єдиний числовий висновок з більшою кількістю входів або предикторів.

У дереві рішень типовим завданням є визначення атрибута на кожному вузлі. Процес називається виділенням атрибутів і має використовувати деякі заходи для ідентифікації атрибута.

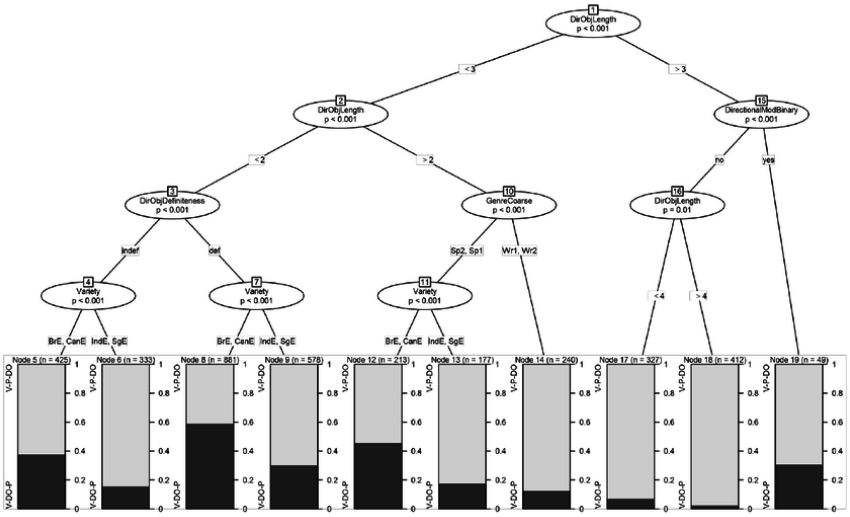



Рис. 1.3. Дерево рішень

**Бегінг**– це ансамблевий метод [3], який використовується у вирішені регресійних задач. У беггінгу використовують один алгоритм, але на його вхід подають випадкові вибірки з набору даних, і дані в цих випадкових вибірках можуть повторюватися. На вхідних даних алгоритм навчається, а потім результат усереднюється.

Найпопулярніший методу бегґінгу – **випадковий ліс**. Як зрозуміло з його назви, він містить велику кількість окремих дерев рішень, які діють як ансамбль. Кожне окреме дерево у випадковому лісі обчислює прогноз класифікації, і клас із найбільшою кількістю голосів стає прогнозом моделі. Тобто ми робимо регресію методом дерев рішень багато разів і обираємо результатом той варіант, який випадав найчастіше.

Основна концепція випадкового лісу проста, але саме через неї модель так добре працює: велика кількість некорельованих дерев рішень, тобто тих, що знаходять рішення незалежно одне від одного й діють спільно, перевершить будь-яке рішення, отримане одним деревом рішення.

До переваг методу можна віднести можливість ефективно обробляти дані з великою кількістю ознак і класів та високу масштабованість. Серед недоліків можна виділити великий розмір моделей, що будуються. Що більша модель, то вища її обчислювальна складність та швидкість знаходження рішень.

Ще один ефективний ансамблевий метод – **бустинг.** Особливістю бустінгу, на відміну від бегінгу, є послідовне, а не паралельне використання алгоритму, до того ж особливу увагу кожного наступного алгоритму звертається на помилки попереднього. Щоб знайти такі помилки, застосовуються алгоритми базового навчання з різним розподілом. Щоразу, коли застосовується базовий алгоритм навчання, він генерує нове слабке правило прогнозування. Це ітеративний, тобто покроковий, процес, і після великої кількості ітерацій бустінг об'єднує ці помилки в одне правило.

Алгоритм навчання штучної **нейронної мережі** , зазвичай званої «нейронною мережею» (НМ), є алгоритмом навчання, віддалено натхненим біологічними нейронними мережами. Обчислення структуруються в термінах взаємозв'язаних груп штучних нейронів, які обробляють інформацію із застосуванням колективістського підходу до обчислень. Сучасні нейронні мережі є нелінійними статистичними інструментами моделювання даних. Їх зазвичай застосовують для моделювання складних взаємозв'язків між входами та виходами, для пошуку закономірностей в даних, або для виявлення статистичної структури в невідомому спільному розподілі ймовірності спостережуваних величин.

Метод **опорних векторів** є набором пов'язаних методів навчання з учителем, які використовуються для класифікації та регресії. Маючи набір тренувальних прикладів, кожен з яких помічено як належний до однієї з двох категорій, алгоритм тренування методу опорних векторів будує модель, яка передбачує, чи новий приклад потрапляє до однієї категорії, чи до іншої.

**Баєсова мережа**, мережа переконань, або спрямована ациклічна графова модель — це ймовірнісна графова модель, яка представляє набір випадкових величин та їхніх умовних незалежностей через спрямований ациклічний граф. Наприклад, баєсова мережа може представляти ймовірнісні взаємозв'язки між хворобами та симптомами. Маючи в розпорядженні симптоми, таку мережу

можна використовувати для обчислення ймовірностей наявності різних хвороб. Існують ефективні алгоритми для виконання висновування та навчання.

Алгоритми LDA і QDA засновані на теоремі Байєса і відрізняються за своїм підходом до класифікації від логістичної регресії. У логістичній регресії можна безпосередньо отримати ймовірність спостереження для класу (Y=k) для конкретного спостереження (X=x). Алгоритм LDA та QDA базується на теоремі Байєса, а класифікація спостереження виконується у два кроки.

LDA (лінійний дискримінантний аналіз) використовується, коли потрібна лінійна межа між класифікаторами, а QDA (квадратичний дискримінантний аналіз) використовується для пошуку нелінійної межі між класифікаторами. LDA і QDA працюють краще, коли класи відповідей роздільні і розподіл X=x для всіх класів є нормальним. Чим більше класів можна розділити і чим більше розподіл є нормальним, тим кращим буде результат класифікації для LDA і QDA.

KNN - метод k-найближчіх сусідів. Метод був вперше розроблений Евеліном Фіксом та Джозефом Лоусоном Ходжесом у 1951 році, і пізніше розвинений Томасом Ковєром.

Метод належить до класу непараметричних, тобто. не вимагає припущень про те, з якого статистичного розподілу була сформована навчальна множина. Отже, класифікаційні моделі, побудовані за допомогою методу KNN, також будуть непараметричними. Це означає, що структура моделі не задається жорстко спочатку, а визначається даними.

Оскільки ознаки, на основі яких проводиться класифікація, можуть мати різну фізичну природу і, відповідно, діапазони значень, для покращення результатів класифікації буде корисно виконати нормалізацію навчальних даних.

Алгоритм KNN можна розділити на дві прості фази: навчання та класифікації. Під час навчання алгоритм просто запам'ятовує вектори ознак спостережень та його мітки класів (тобто. приклади). Також задається параметр

алгоритму kk, який визначає кількість «сусідів», які будуть використовуватися при класифікації.

На фазі класифікації пред'являється новий об'єкт, котрого мітка класу не задана. Він визначається kk найближчих (у сенсі деякої метрики) попередньо класифікованих спостережень. Потім вибирається клас, якому належить більшість kk найближчих прикладів-сусідів, і до цього ж класу відноситься об'єкт, що класифікується.

**Гребнева регресія.** У разі високої колінеарності змінних, стандартна лінійна та поліноміальна регресії стають неефективними. Колінеарність - це відношення незалежних змінних, близьке до лінійного. Наявність високої коллінеарності можна визначити кількома шляхами**:**

1) коефіцієнт регресії не важливий, незважаючи на те, що теоретично змінна повинна мати високу кореляцію з Y;
2) при додаванні або видаленні змінної з матриці X коефіцієнт регресії сильно змінюється;
3) змінні матриці X мають високі попарні кореляції.

Гребенева регресія - це коригуючий міра зниження коллінеарності серед предикторних змінних в регресійній моделі. Коллінеарність - це явище, в якому одна змінна в множинні регресійної моделі може бути передбачена лінійно, виходячи з інших властивостей зі значним ступенем точності. Таким чином, через високу кореляцію змінних, кінцева регресійна модель зведена до мінімальних меж наближеного значення, тобто вона має високу дисперсію.

У **регресії ласо**, як і в гребеневій, ми додаємо умову зміщення у оптимізацію для того, щоб зменшити колінеарність і, отже, дисперсію моделі. Але замість квадратичного зміщення використовується зсув абсолютного значення.

Існує кілька відмінностей між гребеневою регресією та ласо, які відновлюють відмінності у властивостях регуляризацій L2 та L1:

1) вбудований відбір ознак - вважається корисною властивістю, яка є в нормі L1, але відсутня в нормі L2. Відбір ознак є результатом норми L1, що виробляє розріджені коефіцієнти. Наприклад, припустимо, що модель має 100 коефіцієнтів, але лише 10 мають коефіцієнти відмінні від нуля. Відповідно, «решта 90 предикторів є марними у прогнозуванні шуканого значення». Норма L2 виробляє нерозряджені коефіцієнти і може виробляти відбір ознак. Таким чином, можна сказати, що регресія лассо виробляє «вибір параметрів», тому що не вибрані змінні матимуть спільну вагу, що дорівнює 0;
2) розрядженість означає, що незначна кількість вхідних даних у матриці (або векторі) мають значення, відмінне від нуля. Норма L1 виробляє велику кількість коефіцієнтів з нульовим значенням або дуже малі значення з деякими великими коефіцієнтами. Це з попереднім пунктом, у якому зазначено, що лассо виконує вибір властивостей;
3) обчислювальна ефективність: норма L1 немає аналітичного рішення на відміну норми L2. Це дозволяє ефективно вираховувати рішення норми L2. Однак, рішення норми L1 не мають властивості розрядженості, що дозволяє використовувати їх з розрядженими алгоритмами для більш ефективних обчислень.

## 1.2 Огляд існуючих рішень

У роботі доктора, професора, Надера С. Сантаріса та доктора, помічника професора Сінана С. Фаурі Іорданського Приватного Університета Прикладних Наук «Прогнозування вихідної електроенергії електростанції комбінованого типу за допомогою алгоритмів регресії машинного навчання» було використано різні методи регресії, включаючи методи, які були розглянуті у другому розділі МР.

Для прогнозування було використано: лінійну регресію, гребнева регресія, регресія Лассо, еластична мережа, метод випадкового лісу та градієнтний бустинг.

Для визначення якості прогнозування системи автори роботи використовували різні метрики оцінки якості такі, як MSE (середня квадратична похибка), MAPE (середня абсолютна похибка у %), $R^2$(коефіцієнт детермінації), MAE (середня абсолютна похибка), RMSE (середньоквадратична похибка) (Рис. 1.4).

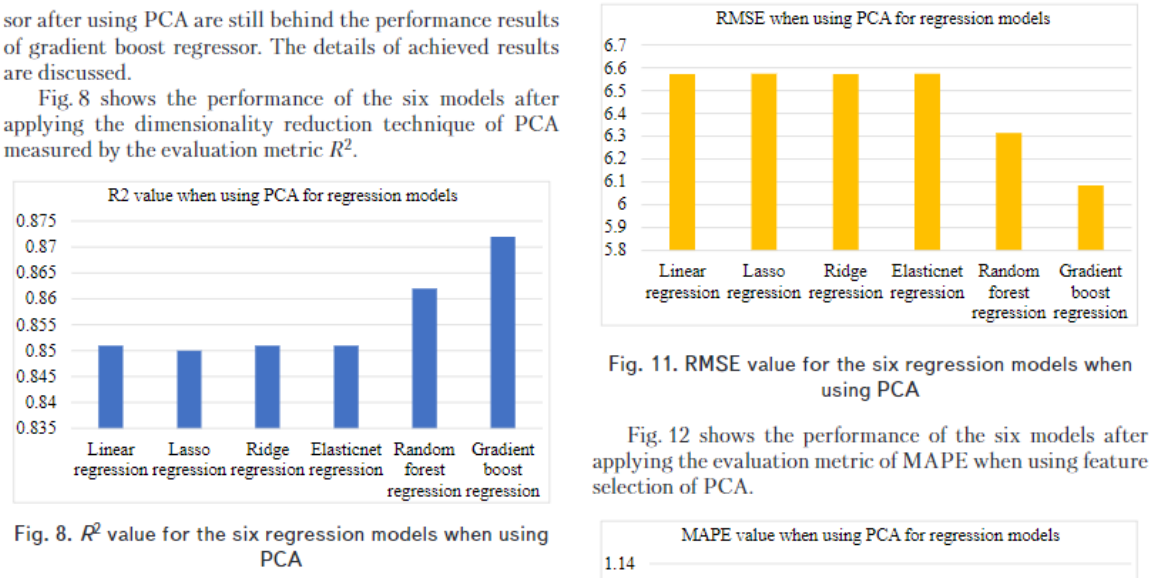



Рис. 1.4. Оцінювання якості прогнозування

У висновках автори порівнюють якість прогнозування різних методів регресії та приводять порівняльну таблицю результатів (Рис. 1.5). Дослідивши порівняльну таблицю можна побачити що найбільш ефективним методом виявився метод градієнтного бустингу, а на другому місці по ефективності опинився метод випадкового дерева.

| Evaluation metric | Linear regression | Lasso regression | Ridge regression | Elastic net regression | Random forest regression | Gradient boost regression |
|---|---|---|---|---|---|---|
| $R^2$ | 0.898 | 0.900 | 0.900 | 0.900 | 0.884 | 0.912 |
| MAE | 4.276 | 4.277 | 4.276 | 4.278 | 4.536 | 3.955 |
| MSE | 28.912 | 28.872 | 28.912 | 28.866 | 33.569 | 25.344 |
| RMSE | 5.376 | 5.373 | 5.376 | 5.372 | 5.793 | 5.034 |
| MAPE (%) | 0.946 | 0.946 | 0.946 | 0.946 | 1.000 | 0.872 |

Рис. 1.5. Порівняльна таблиця результатів

## Висновки до розділу 1

Проаналізувавши предметну сферу, можна сказати, що дана тема МКР є актуальною у нашій країні, оскільки існує реальна проблема енергетики, яку може допомогти вирішити дана інтелектуальна система для прогнозування. У цьому розділі було розглянуто та проаналізовано основні методи прогнозування та оцінки якості прогнозування для машинного навчання, такі як, MSE, RMSE, MAE, MAPE.

Було розглянуто та проаналізовано схожу інтелектуальну систему викладачів Іорданського Приватного Університету Прикладних Наук. У аналогічній роботі було використано 4 методи лінійної регресії та 2 ансамблевих методи (метод градієнтного бустингу та метод випадкового лісу). У порівняльній таблиці аналогічної роботи було приведено оцінки якості прогнозування кожної моделі на тестовому наборі даних. Проаналізувавши результати, автори роботи прийшли до висновку, що найточнішими методами у прогнозуванні стали ансамблеві методи, а саме метод випадкового лісу та градієнтного бустингу. Було проаналізовано результати роботи інтелектуальної системи для подальшого порівняння із власною роботою для визначення якості прогнозування.

# 2 ТЕХНОЛОГІЇ ТА ПІДХОДИ ДО ПРОЕКТУВАННЯ СИСТЕМИ ПРОГНОЗУВАННЯ

## 2.1 Структура інтелектуальної системи

Для проектування інтелектуальної системи для початку необхідно створити її структуру та визначити етапи проектування (Рис. 2.1).

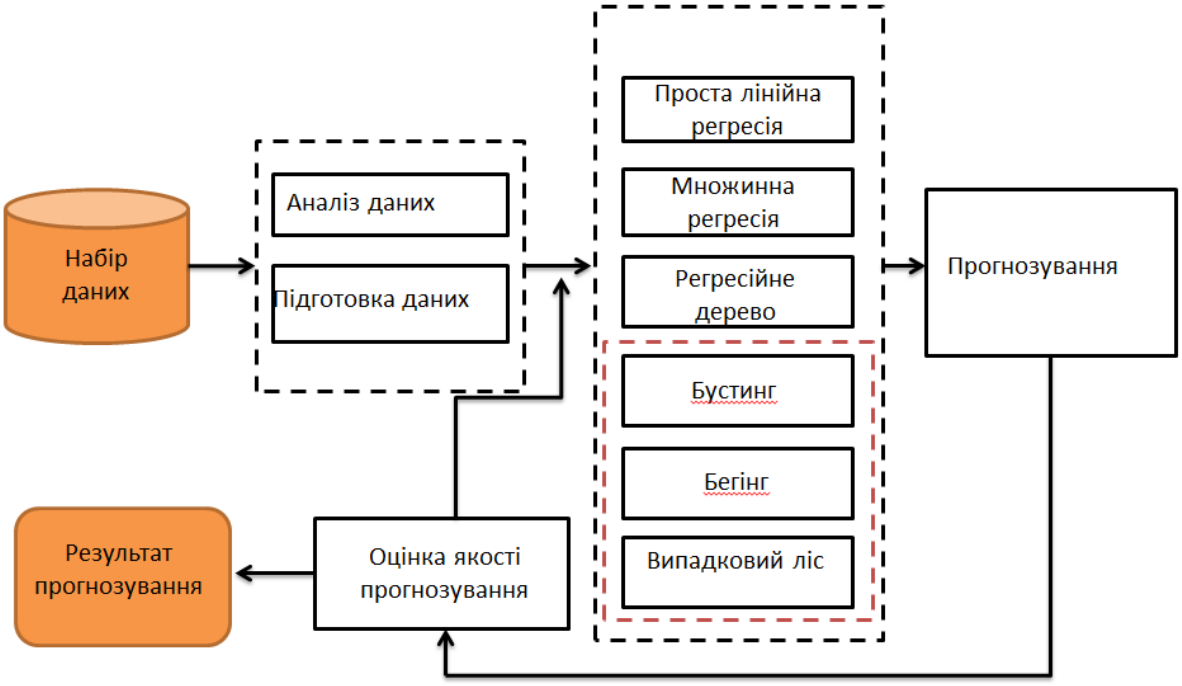



Рис. 2.1. Структура інтелектуальної системи

Проаналізувавши створену структуру інтелектуальної системи, можна розділити проектування на декілька етапів:

1) перший етап – завантаження даних, їх попередній аналіз та підготовка до прогнозування (переведення до однакового типу, видалення пропущених значень, дослідження кореляції між незалежними змінними та міткою);
2) другий етап – створення методів регресії та прогнозування на тренувальному наборі даних.

3) третій етап – тестування методів регресії на контрольному наборі даних.
4) Четвертий етап – оцінка якості прогнозуваннясистеми, використовуючи різні метрики оцінки якості прогнозування (MSE, RMSE, MAE, MAPE).

## 2.2 Обрання технології розробки системи

Для проектування інтелектуальної системи для прогнозування вихідної електроенергії електростанції комбінованого типу розглядалися дві мови програмування – це R та Python.

R — це мова програмування для статистичних обчислень і графіки, яка підтримується командою R Core і R Foundation for Statistical Computing. Створений статистиками Россом Іхакою та Робертом Джентльменом, R використовується серед майнерів даних і статистиків для аналізу даних та розробки статистичного програмного забезпечення. Користувачі створили пакети для розширення функцій мови R.

R – платформо незалежний, він добре інтегрується з іншими мовами програмування. Поряд із аналізом даних, R пристосований для візуалізації даних.

Незважаючи на відносну простоту інтеграції з іншими інструментами, R має ряд особливостей, які ускладнюють його вивчення. До них, наприклад, можна віднести нетрадиційні структури даних та індексування (яке починається з 1 замість 0).

R менш популярний, ніж Python, тому масив документації, необхідний розробникам для створення програм в області МН, менше.

Python – це високорівнева мова програмування, яка має безліч різних способів застосування, включаючи науку про дані та внутрішню веб-розробку. Він був створений Python Foundation на початку 1990-х і є потужним інструментом для аналізу даних, що широко використовується в технології

великих даних. Особливого статусу йому надає численна спільнота розробників МН, яка в основному зосереджена на швидкозростаючому ІІ-напрямку.

Завдяки активній спільноті для Python з'явилося багато готових бібліотек МН. Ця мова — платформо незалежна, тому її можна адаптувати практично до будь-якої операційної системи. Ще одна перевага Python пов'язана з його відкритістю - він побудований на базі технологій Open Source.

Що стосується мінусів Python, то оскільки це динамічно типізована мова, робота з нею у середовищі МН може викликати проблеми. Однією з них є складність відстеження помилок у коді, що пов'язано з розростанням кодової бази програми та, відповідно, з її складністю. У деяких випадках (складні проекти для великих організацій) аудит коду може виливатися у значні фінансові витрати, забирати багато часу та впливати на продуктивність проекту.

Зваживши усі переваги та недоліки двох мов програмування, було вирішено розробляти інтелектуальну систему на базі мови програмування R.

Завантажити R можна з офіційного сайту (Рис. 2.2), який називається CRAN (TheComprehensiveRArchiveNetwork). Також з цього сайту можна встановлювати усі необхідні пакети. Кожен пакет містить набір власних методів, тому для використання певного методу необхідно спочатку завантажити необхідний пакет, а потім завантажити до своєї середи розробки. Також більшість пакетів мають свою унікальну базу даних, яку можна використовувати для тестування різних методів класифікації та регресії.

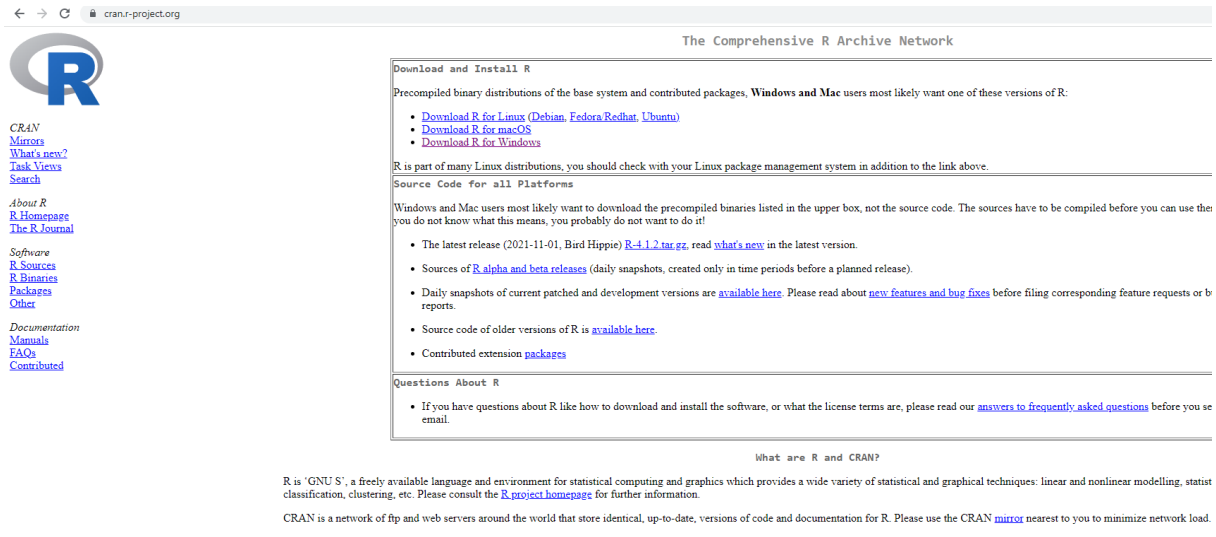


Рис. 2.2. Сервер CRAN

Порівнюючи із мовою програмування Python, R більш складний у вивчені, але кращий коли потрібно побудувати різні графіки, необхідні для порівняльної статистики.

RStudio — це інтегроване середовище розробки (IDE) длямови програмування R(Рис.2.3). Він доступний у двох форматах: RStudio Desktop — це звичайна настільна програма, а RStudio Server працює на віддаленому сервері та дозволяє отримати доступ до RStudio за допомогою веб-браузера.

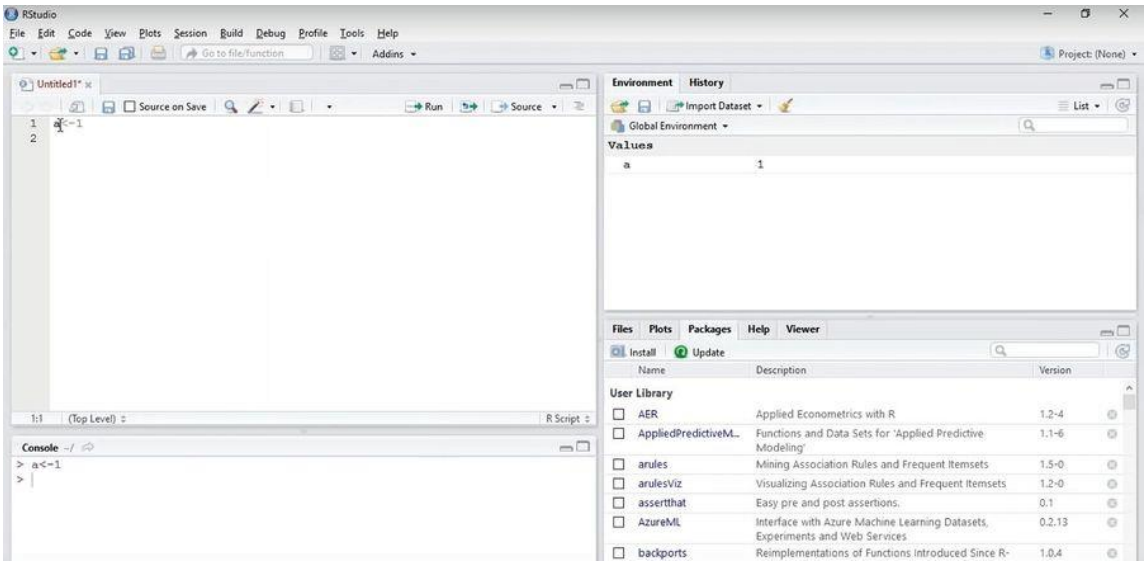



Рис. 2.3. Інтерфейс CP RStudio

## 2.3. Методи машинного навчання

Першу програму на основі алгоритмів, здатних самонавчатися, розробив Артур Самуель (Arthur Samuel) в 1952 році, призначена вона була для гри в шашки. Самуель дав і перше визначення терміну «машинне навчання»: це «область досліджень розробки машин, які не є заздалегідь запрограмованими». Більш точне визначення терміну «навчання» дав набагато пізніше Т. М. Мітчелл: кажуть, що комп'ютерна програма навчається на основі досвіду E по відношенню до деякого класу задач T і заходи якості P,

якщо якість вирішення завдань з T, виміряний на основі P, поліпшується з набуттям досвіду E.

Вже в 1957 році була запропонована перша модель нейронної мережі, що реалізує алгоритми машинного навчання, схожі на сучасні. В даний час ведеться розробка самих різних систем машинного навчання, призначених для використання в таких технологіях майбутнього, як Інтернет Речей, Промисловий Інтернет Речей, в концепції «розумний» місто, при створенні безпілотного транспорту і в багатьох інших.

Машинне навчання, реорганізоване як окрема область, почало бурхливо розвиватися в 1990-х роках. Ця область змінила свої цілі з досягання штучного інтелекту на розв'язання розв'язних задач практичного характеру. Вона змістила фокус із символьних підходів, успадкованих нею від ШІ, в бік методів та моделей, позичених зі статистики та теорії ймовірності. Вона також виграла від збільшуваної доступності оцифрованої інформації та можливості розповсюджування її через Інтернет.

Машинне навчання та добування даних часто використовують одні й ті ж методи, і значно перекриваються, але в той час як машинне навчання фокусується на передбаченні на основі *відомих* властивостей, вивчених з тренувальних даних, добування даних фокусується на відкритті невідомих(раніше) властивостей даних (це є кроком аналізу з відкривання знань у базах даних). Добування даних використовує багато методів машинного навчання, але з іншими цілями; з іншого боку, машинне навчання також застосовує методи добування даних як «навчання без учителя», або як крок попередньої обробки для покращення точності механізму навчання. Велика частина плутанини між цими двома дослідницькими спільнотами (які часто мають окремі конференції та окремі журнали, з ECML PKDD як основним винятком) виходить з основних припущень, з якими вони працюють: у машинному навчанні продуктивність зазвичай оцінюється по відношенню до здатності *відтворювати відоме* знання, тоді як у відкриванні знань та добуванні

даних ключовою задачею є відкривання *не відомого* раніше знання. При оцінці по відношенню до відомих знань неінформований (некерований) метод легко програватиме іншим керованим методам, тоді як у типовій задачі *KDD* керовані методи застосовуватися не можуть в силу відсутності тренувальних даних.

Машинне навчання також має тісні зв'язки з оптимізацією: багато задач навчання формулюються як мінімізація деякої функції втрат на тренувальному наборі прикладів. Функції втрат виражають розбіжність між передбаченнями тренованої моделі та дійсними зразками задачі (наприклад, у класифікації потрібно призначати мітки зразкам, і моделі тренуються правильно передбачати попередньо призначені мітки набору прикладів). Різниця між цими двома областями виникає з мети узагальнення: в той час як алгоритми оптимізації можуть мінімізувати втрати на тренувальному наборі, машинне навчення зосереджене на мінімізації втрат на небачених зразках.

Про те, що на машинне навчання зараз покладають великі надії, свідчать такі факти:

1) В компанії Google вважають, що скоро її продукти «перестануть бути результатом традиційного програмування — в їх основу буде покладено машинне навчання».
2) Компанії Google, Facebook, Apple, Amazon, Microsoft і китайська фірма Baidu вступили в боротьбу за талановитих фахівців у сфері штучного інтелекту;
3) Марк Цукерберг, генеральний директор Facebook, особисто — по телефону і по відеочату — бере участь в спробах його компанії переманити найкращих випускників.
4) Відвідуваність на найважливіших академічних конференціях в цій сфері збільшилася майже в чотири рази.
5) Такі нові продукти, як Siri від Apple, M від Facebook, Echo від Amazon були створені за допомогою машинного навчання.

У найзагальнішому випадку розрізняють два типу машинного навчання: навчання по прецедентах, або індуктивне навчання, і дедуктивне навчання. Оскільки останнє прийнято відносити до області експертних систем, то терміни «машинне навчання» і «навчання по прецедентах» можна вважати синонімами. Цей метод навчання зараз, як прийнято говорити, в тренді, а ось експертні системи переживають кризу. Бази знань, що лежать в їх основі, важко узгоджувати з реляційною моделлю даних, тому промислові СУБД неможливо ефективно використовувати для наповнення баз знань експертних систем.

Навчання по прецедентах, в свою чергу, поділяють на три основних типи: контрольоване навчання, або навчання з учителем (supervised learning), неконтрольоване навчання (unsupervised learning), або навчання без учителя, і навчання з підкріпленням (reinforcement learning).

Крім названих, розробляються і інші методи навчання: активне, багатозадачне, різноманітне, трансферне і т.д. Особливо успішно розвивається в останні роки «глибоке навчання», при використанні якого можуть успішно поєднуватися алгоритми навчання з вчителем і без вчителя.

Існує кілька методів, які впливають на просування систем автоматичного навчання та вдосконалення відповідно до досвіду. Але вони підпадають під різні категорії чи типи, такі як наглядове навчання, непідконтрольне навчання, зміцнення навчання, представницьке навчання тощо. Нижче наведено методи, які підпадають під машинне навчання:

1) **регресія**; алгоритми регресії в основному використовуються для прогнозування чисел, тобто коли вихідний сигнал є реальним або безперервним значенням. Оскільки вона підпадає під наглядове навчання, вона працює з навченими даними для прогнозування нових тестових даних. Наприклад, вік може бути суцільним значенням, оскільки він збільшується з часом;

2) **класифікація**; класифікаційна модель, метод контрольованого навчання, робить висновок із спостережуваних значень як одного або декількох результатів у категоричній формі. Наприклад, електронна пошта має фільтри, такі як папка "Вхідні", "Чернетки", "Спам". У моделі класифікації існує ряд алгоритмів, таких як Логістична регресія, Дерево рішень, Випадковий ліс, Багатошарове сприйняття тощо. У цій моделі дані класифікуються спеціально і їм призначається відповідно до цих класів мітки. Класифікатори бувають двох типів: двійкові класифікатори - класифікація з двома окремими класами та двома вихідними; багатокласний класифікатор s - класифікація з більш ніж 2 класами;
3) **кластеризація** - це технологія машинного навчання, яка включає класифікацію точок даних на конкретні групи. Якщо є деякі об'єкти або точки даних, можна застосувати алгоритм кластеризації для аналізу та групування їх відповідно до їх властивостей та особливостей. Цей метод непідконтрольної методики застосовується через його статистичні прийоми. Алгоритми кластерів роблять прогнози на основі даних про навчання та створюють кластери на основі подібності чи незнайомості.

   Методи кластеризації:

   - методи на основі щільності - у цьому методі кластери вважаються щільними регіонами залежно від їх подібності та відмінності від нижньої щільної області;
   - ієрархічні методи - кластери, утворені цим методом, є деревоподібними структурами. Цей метод формує дерева або кластери з попереднього кластера. Існує два типи ієрархічних методів: агломераційний (підхід знизу вгору) та подільний (підхід зверху вниз);
   - методи розподілу - цей метод розділяє об'єкти на основі k-кластерів і кожен метод утворює єдиний кластер;

- методи на основі Гріса - у цьому методі дані об'єднуються в ряд комірок, які утворюють структуру, подібну до сітки;

4) **виявлення аномалії** - це процес виявлення несподіваних елементів або подій у наборі даних. Деякі сфери, де використовується ця методика, - це виявлення шахрайства, виявлення несправностей, моніторинг здоров'я системи тощо. Виявлення аномалії може бути класифіковано як:

- аномалії точок - аномалії точок визначаються, коли окремі дані несподівані;
- контекстуальні аномалії - коли аномалії є контекстними, то це називається контекстуальними аномаліями;
- колективні аномалії - коли колекція або група пов'язаних елементів даних є аномальною, то це називається колективною аномальною.

Існують певні методики виявлення аномалії наступним чином:

1) статистичний метод; він допомагає виявити аномалії шляхом вказівки даних, що відхиляються від статистичних методів, таких як середнє значення, медіана, режим тощо;
2) виявлення аномалії на основі щільності : базується на алгоритмі k-найближчого сусіда.
3) алгоритм аномалії на основі кластеризації - точки даних збираються як кластер, коли вони підпадають під одну групу і визначаються з локальних центроїдів.
4) супер векторна машина - алгоритм тренує себе для кластеризації звичайних екземплярів даних та визначає аномалії, використовуючи дані тренувань.

## 2.4. Огляд обраних методів регресії для прогнозування

Для машинного навчання використовують різні технології та алгоритми. Зокрема, можуть застосовуватися дискримінантний аналіз, байєсовські класифікатори та багато інших математичних методів. Але в кінці XX століття все більше уваги почали приділяти штучним нейронним мережам

(ANN). Черговий вибух інтересу до них почався в 1986 році, після істотного розвитку т.зв. «Методу зворотного поширення помилки», який з успіхом застосували при навчанні нейронної мережі.

ANN є системою з'єднаних і взаємодіючих між собою штучних нейронів, виконаних на основі порівняно простих процесорів. Кожен процесор ANN періодично отримує сигнали від одних процесорів (або від сенсорів, або від інших джерел сигналів) і періодично посилає сигнали іншим процесорам. Всі разом ці прості процесори, з'єднані в мережу, здатні вирішувати досить складні завдання.

Найчастіше нейрони розташовуються в мережі за рівнями (їх ще називають шарами). Нейрони першого рівня — це, як правило, вхідні. Вони отримують дані ззовні (наприклад, від сенсорів системи розпізнавання осіб) і після їх обробки передають імпульси через синапси нейронів на наступному рівні. Нейрони на другому рівні (його називають прихованим, оскільки він безпосередньо не пов'язаний ні з входом, ні з виходом ANN) обробляють отримані імпульси і передають їх нейронам на вихідному рівні. Оскільки мова йде про імітацію нейронів, то кожен процесор вхідного рівня пов'язаний з декількома процесорами прихованого рівня, кожен з яких, в свою чергу, пов'язаний з декількома процесорами рівня вихідного. Така архітектура найпростішої ANN, яка здатна до навчання і може знаходити прості взаємозв'язку в даних.

Глибоке (глибинне) навчання може бути застосоване лише по відношенню до більш складних ANN, що містить кілька прихованих рівнів. При цьому рівні нейронів можуть чергуватися з шарами, які виконують складні логічні перетворення. Кожен наступний рівень мережі шукає взаємозв'язки в попередньому. Така ANN здатна знаходити не тільки прості взаємозв'язки, а й взаємозв'язки між взаємозв'язками. Саме завдяки переходу на нейромережу з глибинним навчанням компанії Google вдалося різко підвищити якість роботи свого популярного продукту «Перекладач». Зокрема,

якість перекладу між англійською та французькою мовами підвищився відразу на 7 балів, тобто більш ніж на 20%. Попередня система, яка виконувала фразовий статистичний машинний переклад, домоглася подібного поліпшення за весь час свого існування (з 2006 року).

**Проста лінійна регресія** використовується для оцінки зв'язку між двома кількісними змінними[5].Можна використовувати просту лінійну регресію, коли необхідно дізнатись:

1) наскільки сильний зв'язок між двома змінними;
2) значення залежної змінної при певному значенні незалежної змінної.

Ми також можемо використовувати регресію, щоб передбачити значення змінної відповіді на основі значень важливих предикторів. Це зазвичай називають прогнозним моделюванням. Або ми можемо використовувати регресійні моделі для оптимізації, щоб визначити параметри факторів для оптимізації відповіді.

Оскільки проста лінійна регресія прогнозує значення на основі одного незалежного предиктора, якість прогнозування такої моделі буде нижча ніж при використанні множинної регресії або регресійних дерев.

**Множина лінійна регресія** відноситься до статистичної техніки, яка використовується для прогнозування результату змінної на основі значення двох або більше змінних. Іноді її називають просто множинною регресією, і це розширення лінійної регресії [6].

Модель прогнозування можна побудувати на основі усіх предикторів, декількох (які найбільш корелюють із залежною змінною) або використовуючи метод найменших квадратів.

Загальна формула множинної лінійної регресії має вигляд:

$$y_i = \beta_0 + \beta_1 \cdot x_{i1} + \beta_2 \cdot x_{i2} + \ldots + \beta_n \cdot x_{in} + \epsilon, \text{ де:}$$

$y_i$ – це залежна або прогнозована змінна;

$\beta_0$ – точка перетину з віссю y;

$\beta_1$ та $\beta_2$ – коефіцієнти регресії, представляючі зміну у відносно зміни на одну одиницю $x_{i1}$ та $x_{i2}$відповідно;

$\beta_n$ – коефіцієнт нахилу для кожної незалежної змінної;

$\epsilon$ – випадкова похибка (залишок) моделі.

Оскільки множинна регресія прогнозує на основі декількох предикторів, точність прогнозування такої моделі буде значно вищою ніж простої лінійної регресії.

**Дерево регресії** будується за допомогою процесу, відомого як бінарне рекурсивне розбиття, яке представляє собою ітеративний процес, який розбиває дані на розділи або гілки, а потім продовжує розбивати кожен розділ у більш дрібних групах по мірі просування методу вгору по кожній гілці [7].

Спочатку всі записи в навчальному наборі групуються в один і той же розділ. Потім алгоритм починає розподіляти дані по першим двом розділам або гілкам, використовуючи можливе подвійне розділення для кожного поля. Алгоритм вибирає розділення, яке мінімізує суму квадратів, відклонених від середнього значення в двох окремих розділах. Потім це правило розділення застосовується до кожної з нових гілок. Цей процес триває до тех пір, поки кожен вузол не досягає заданого використання мінімального розміру вузла і не стане кінцевим. Якщо сумма квадратів відхилень від середнього увузлі рівне нулю, то цей вузол вважається кінцевимвузлом, навіть якщо він не досягає мінімального розміру.

Оскільки дерево виросло з тренувального набору, повністю розвинене дерево зазвичай страждає від надмірної підгонки. Це перенасичення призводить до поганої продуктивності реальних даних. Тобто, дерево необхідно обрізати за допомогою набору перевірки.

Дерево обрізається, щоб мінімізувати суму:

1) дисперсії вихідної змінної в даних перевірки, взятих по одному кінцевому вузлу за раз;

2) добуток стійкого фактора складності та кількості кінцевих вузлов. Якщо коефіцієнт складності вартості вказано рівним нулю, то скорочення — це просто пошук дерева, яке краще всього працює на даних перевірки з точки зору загальної дисперсії кінцевих вузлів. Більші значення коефіцієнта складності витрат приводять до менших дерев. Відрізання виконується за принципом «останній прийшов — перший пішов», тобто останній створеннийвузол першим відрізається.

**Ансамблеві методи** дозволяють нам комбінувати декілька моделей дерев слабкої регресії, які утворюють нову точну модель дерева сильної регресії. Ці методи працюють, створюючи кілька різноманітних моделей регресії, беручи різні зразки вихідного набору даних, потім об'єднуючи їх вихідні дані.

Ця комбінація моделей ефективно знижує дисперсію у сильній моделі. Три типи ансамблевих методів(бегінг, бустинг і метод випадковоголісу), розрізняються за трьома пунктами:

1) вибір навчального набору для кожного предиктора або слабкої моделі;

2) як генеруються слабкі моделі;

3) як результати об'єднуються.

У всіх трьох методах кожна слабка модель навчається на всьому тренувальному наборі, щоб освоїти якусь частину набору даних.

**Бегінг** створює кілька навчальних наборів, використовуючи випадкову вибірку із заміною (самозавантажувальна вибірка), застосовує алгоритм дерева регресії до кожного набору даних, потім бере середнє серед моделей для розрахунку прогнозів нових даних. Найбільшою перевагою бегінга є відносна простота розділу алгоритму, що робить його найкращим вибором для великих наборів даних.

Підвищення створює сильну модель шляхом послідовного навчання моделей, щоб зосередитись на записах, які отримують неточні передбачені значення у попередніх моделях. Після завершення всі предиктори об'єднуються зваженою більшістю голосів.

**Метод випадкового лісу**– різновид бегінга. Цей метод працює шляхом навчання кількох дерев слабкої регресії з використанням фіксованої кількості випадково вибраних ознак ($\sqrt{p}$ для класифікації та p/3 для передбачення), потім бере середнє значення для слабких учнів. Як правило, кількість згенерованих слабких дерев може варіюватися від кількох сотень до кількох тисяч залежно від розміру та складності навчальної вибірки. Випадкові дерева можна розділити, тому що вони є варіантом бегінга. Проте, оскільки випадкові дерева вибирають обмежену кількість ознак кожної ітерації, продуктивність випадкових дерев вище, ніж в бегінга.

**Бустинг –** це один із методів ансамблю, який на відміно від бегінга виконує навчання послідовно, тобто кожне наступне дерево вчиться на помилці попереднього, в результаті чого утворюється сильне дерево, коли бегінг навчає дерева паралельно один від одного, а потім усереднює загальну помилку. Недолік методу бустингу полягає у повільності будування регресійної моделі.

**Поліноміальна регресія.** Для створення такої моделі, яка підійде для даних, що нелінійно розділяються, можна використовувати поліноміальну регресію. У цьому методі проводиться крива лінія, залежна від точок площини. У поліноміальній регресії рівень деяких незалежних змінних перевищує 1.

Деякі змінні мають ступінь, інші — ні. Також можна вибрати певну міру для кожної змінної, але для цього необхідні певні знання про те, як вхідні дані пов'язані з вихідними. Порівняйте лінійну та поліноміальну регресію нижче.

Декілька важливих пунктів про поліноміальну регресію:

1) моделює нелінійно розділені дані (чого може лінійна регресія). Вона гнучкіша і може моделювати складні взаємозв'язки.

2) повний контроль за моделюванням змінних об'єкта (вибір ступеня).

3) потрібно уважно створювати модель. Необхідно мати деякі знання про дані, для вибору найбільш відповідного ступеня.

4) при неправильному виборі ступеня дана модель може бути перенасичена.

Еластична мережа – це гібрид методів регресії ласо та гребеневої регресії. Вона використовує як L1, і L2 регуляризації, враховуючи ефективність обох методів.

Практичною перевагою використання регресії лассо та гребеневої регресії є те, що це дозволяє еластичній мережі успадковувати деяку стабільність гребеневої регресії при обертанні.

Декілька важливих пунктів про регресію еластичної мережі:

1) вона створює умови для групового ефекту при високій кореляції змінних, а не обнуляє деякі з них як метод ласо.
2) немає обмежень щодо кількості вибраних змінних.

Коли зв'язок між незалежною змінною та змінною ефекту не зрозумілий, узагальнена адитивна модель зазвичай може використовуватися для визначення того, чи має змінна відношення нелінійний зв'язок.

GAM бере кожну передикторну змінну моделі і ділить її на кілька частин (розділених вузлом), а потім підганяє поліноміальну функцію до кожної частини окремо. Принцип GAM полягає в тому, щоб мінімізувати залишки (ступінь відповідності) за максимальної простоти (мінімально можливі ступені свободи). Деякі або всі незалежні змінні у регресійній моделі використовують функції згладжування для зниження ризику моделі, викликаного лінійним налаштуванням. Припущення моделі не є суворим. Наприклад, немає необхідності припускати, що незалежні змінні лінійно пов'язані із залежною змінною (лінійною або нелінійною). І щоб вирішити логістичну регресію, коли кількість незалежних змінних велика, легко викликати прокляття розмірності (прокляття розмірності).

Проаналізувавши усі методи регресії, які будуть використовуватися під час розробки інтелектуальної системи, можна зробити висновок, що

найякісніша модель буде побудована при використанні ансамблевих методів, а саме: бустингу, бегінгу та методу випадкового лісу, що є виключним випадком, коли у побудові дерев беруть участь усі предиктори.

Найнижчу якість прогнозування повинна продемонструвати модель простої лінійної регресії. Множинна регресія продемонструє кращий результат за просту лінійну регресію, але все одно гірше ніж ансамблеві методи.

## 2.5 Опис обраних метрик оцінки якості прогнозування

Для оцінки якості прогнозування було обрано такі метрики: MSE, RMSE, MAE, MAPE.

**MSE (Mean Squared Error)** – середньоквадратична похибка. Вимірює усереднення квадратів похибок — тобто, середнє квадратичної різниці між оцінками значень та справжнім значенням. MSE застосовується у ситуаціях, колинеобхіднопідкреслит великі помилки та вибрати модель, яка дає менше великих помилок прогнозування. Грубі помилки стають помітнішими за рахунок того, що помилка прогнозу зводиться в квадрат. Модель, яка дає менше значення середньоквадратичної помилки, має менше грубих помилок.

**RMSE (Root MeanSquaredError) –** середньоквадратична похибка. RMSE– це фактично $\sqrt{MSE}$. Тобто ця метрика має пряму залежність із середньоквадратичною похибкою.

**MAE (MeanAbsoluteError)** – є мірою помилок між парними спостереженнями, що виражають одне й те саме явище.MAE– це середнє арифметичне абсолютних похибок спрогнозованого значення та фактичного:

$$|e_i| = |y_i - x_i|, \text{ де:}$$

$|e_i|$ – абсолютна похибка;

$y_i$ – спрогнозоване значення;

$x_i$ – фактичне значення.

**MAPE (Mean Absolute Percentage Error) –** середня абсолютна похибка у відсотках. MAPEфактично це MAE,але у відсотках. Недоліком даної метрики є

те, що якщо фактичне значення дуже мале, то значення оцінки буде прагнути до нескінченності.

Для використання цих метрик для оцінки якості прогнозування у середовищі розробки RStudio необхідно завантажити та встановити пакет Metrics.

Для застосування метрик необхідно визвати їх відповідними функціями: mse(actual, predicted). Усі чотири функції у якості параметру приймають фактичне значення та прогнозоване.

## 2.6 Опис набору даних для прогнозування

Для прогнозування вихідної електроенергії електростанції комбінованого типу було взято набір даних **Combined Cycle Power Plant** з репозиторію для машинного навчання UCI, який є загальнодоступний[4]. Мета цього датасету в прогнозуванні вихідної чистої електричної енергії електростанції комбінованого циклу протягом 2006-2011 років на основі показників чинників навколишнього середовища, які збиралися за допомогою спеціальних датчиків та приладів, які знаходились по периметру усієї електростанції. Увесь набір отримано з реальних спостережень.

Цей набір складається з 9568 спостережень, кожне з яких має 4 атрибути та мітку. Список усіх атрибутів разом з міткою:

1) температура (T) в діапазоні 1,81 ° C і 37,11 ° C.
2) тиск навколишнього середовища (AP) в діапазоні 992,89-1033,30 мілібар.
3) відносна вологість (RH) в діапазоні від 25,56% до 100,16%.
4) витяжний вакуум (V) в діапазоні 25,36-81,56 см рт.
5) чистий погодинний вихід електроенергії (EP) 420,26-495,76 МВт.

Даний набір даних особливо підходить для прогнозування, оскільки містить 9568 спостережень, що дуже сильно підвищує точність моделі прогнозування.

## Висновки до розділу 2

У цьому розділі був проведений аналіз структури інтелектуальної системи для прогнозування вихідної електроенергії електростанції комбінованого типу. Проаналізувавши структуру системи, проектування було розділене на 4 етапи.

Також було проаналізовано обрану мову програмування та середовища розробки системи та визначено особливості та переваги в порівнянні із іншими мовами програмування.

Було розглянуто та проаналізовано обраний для розробки системи датасет, який містить 9568 спостережень.

Було визначено та описано усі методи регресії, які використовуються при прогнозуванні. Ознайомлено із метриками оцінки якості прогнозування, такими як, MSE, RMSE, MSE, MAPE.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗУВАННЯ ВИХІДНОЇ ЕЛЕКТРОЕНЕРГІЇ ЕЛЕКТРОСТАНЦІЇ КОМБІНОВАНОГО ТИПУ

## 3.1 Завантаження та підготовка даних

Спочатку необхідно завантажити файл даних з розширенням .xlsx на ПК, після чого у середовищі RStudio необхідно імпортувати файл з даними (Рис. 3.1). Після завантаження ми можемо продивитися дані в табличному форматі. Перевіряємо правильність завантаження продивляючись зміну dataset з панелі джерел даних (Рис. 3.2).

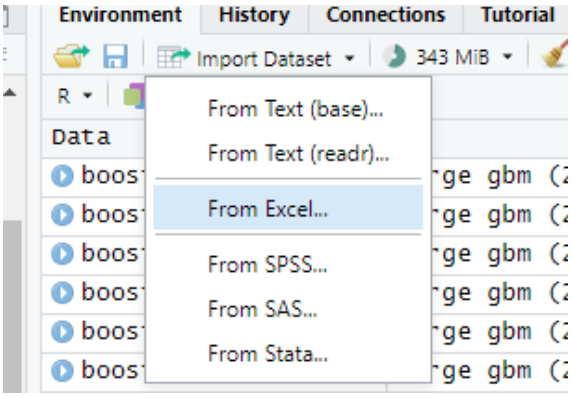
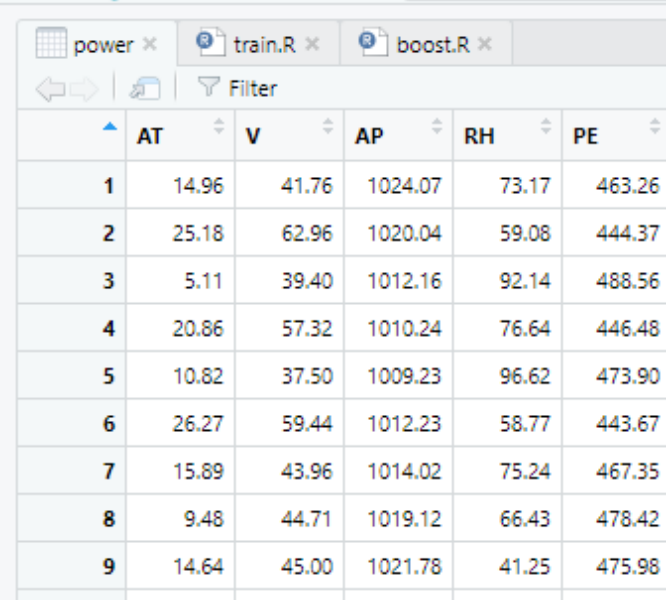



Рис. 3.1. Імпортування файлу данних




| | AT | V | AP | RH | PE |
|---|---|---|---|---|---|
| 1 | 14.96 | 41.76 | 1024.07 | 73.17 | 463.26 |
| 2 | 25.18 | 62.96 | 1020.04 | 59.08 | 444.37 |
| 3 | 5.11 | 39.40 | 1012.16 | 92.14 | 488.56 |
| 4 | 20.86 | 57.32 | 1010.24 | 76.64 | 446.48 |
| 5 | 10.82 | 37.50 | 1009.23 | 96.62 | 473.90 |
| 6 | 26.27 | 59.44 | 1012.23 | 58.77 | 443.67 |
| 7 | 15.89 | 43.96 | 1014.02 | 75.24 | 467.35 |
| 8 | 9.48 | 44.71 | 1019.12 | 66.43 | 478.42 |
| 9 | 14.64 | 45.00 | 1021.78 | 41.25 | 475.98 |

Рис. 3.1. Панель джерел даних

Використовуємо команди head та tail для отримання відповідно перших і останніх даних з dataset (Рис.3.3 – Рис. 3.4).

```
> head(power)
      AT     V       AP    RH     PE
1 14.96 41.76 1024.07 73.17 463.26
2 25.18 62.96 1020.04 59.08 444.37
3  5.11 39.40 1012.16 92.14 488.56
4 20.86 57.32 1010.24 76.64 446.48
5 10.82 37.50 1009.23 96.62 473.90
6 26.27 59.44 1012.23 58.77 443.67
```

Рис. 2.3. Результат виконання head

```
> tail(power)
         AT     V       AP    RH     PE
9563 14.02 40.10 1015.56 82.44 467.32
9564 16.65 49.69 1014.01 91.00 460.03
9565 13.19 39.18 1023.67 66.78 469.62
9566 31.32 74.33 1012.92 36.48 429.57
9567 24.48 69.45 1013.86 62.39 435.74
9568 21.60 62.52 1017.23 67.87 453.28
>
```

Рис. 3.3. Результат виконання tail

Далі використовуємо функції summary для отримання базової статистики по кожному атрибуту (Рис.3.5).

```
> summary(power)
       AT              V               AP              RH              PE
 Min.   : 1.81   Min.   :25.36   Min.   : 992.9   Min.   : 25.56   Min.   :420.3
 1st Qu.:13.51   1st Qu.:41.74   1st Qu.:1009.1   1st Qu.: 63.33   1st Qu.:439.8
 Median :20.34   Median :52.08   Median :1012.9   Median : 74.97   Median :451.6
 Mean   :19.65   Mean   :54.31   Mean   :1013.3   Mean   : 73.31   Mean   :454.4
 3rd Qu.:25.72   3rd Qu.:66.54   3rd Qu.:1017.3   3rd Qu.: 84.83   3rd Qu.:468.4
 Max.   :37.11   Max.   :81.56   Max.   :1033.3   Max.   :100.16   Max.   :495.8
>
```

Рис. 3.4. Результат виконання summary

Наступний крок – аналіз структури даних та преведення її до необхідного типу.  Для цього використовуємо функцію str (Рис.3.6)

```
> str(power)
Classes 'tbl_df', 'tbl' and 'data.frame':       9568 obs. of  5 variables:
 $ AT: num  14.96 25.18 5.11 20.86 10.82 ...
 $ V : num  41.8 63 39.4 57.3 37.5 ...
 $ AP: num  1024 1020 1012 1010 1009 ...
 $ RH: num  73.2 59.1 92.1 76.6 96.6 ...
 $ PE: num  463 444 489 446 474 ...
> 
```

Рис. 3.5. Результат виконання str

Після виконання підготовки даних та їх аналізу можна сказати, що дані знаходяться у потрібному вигляді: назви рядків міняти не потрібно, усі дані знаходяться у числовому типі, пустих значень немає.

### 3.2 Знаходження взаємозв'язків між предикторами

Спочатку потрібно побудувати діаграми розподілу (Рис. 3.7). Функція автоматично конвертує логічний та факторний тип даних в числовий, для відображення даних в вигляді діаграм.

Код:

*scattrplotMatrix(power)*

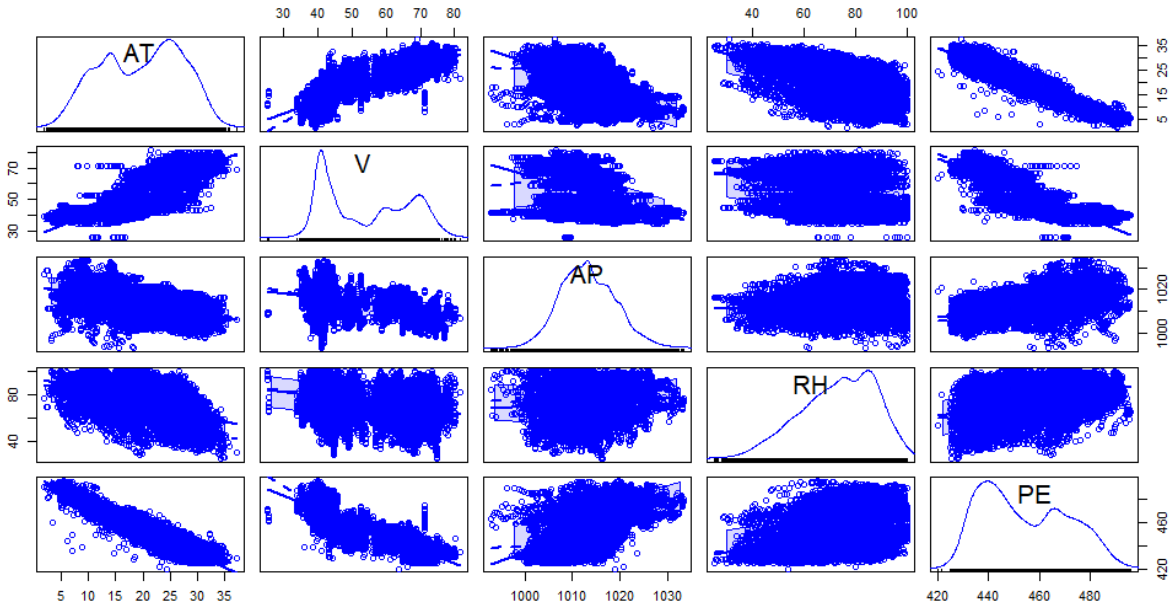



Рис. 3.7. Діаграми розподілу

Наступний крок – дослідження кореляції в даних (Рис.3.8). Кореляція – процес знаходження взаємозалежностей між даними.

Код:

*cor(power)*

```
> cor(power)
           AT          V          AP          RH          PE
AT  1.0000000  0.8441067 -0.50754934 -0.54253465 -0.9481285
V   0.8441067  1.0000000 -0.41350216 -0.31218728 -0.8697803
AP -0.5075493 -0.4135022  1.00000000  0.09957432  0.5184290
RH -0.5425347 -0.3121873  0.09957432  1.00000000  0.3897941
PE -0.9481285 -0.8697803  0.51842903  0.38979410  1.0000000
> 
```

Рис. 3.8. Кореляція даних

## 3.3 Проста регресійна модель

Для створення простої регресійної моделі використовується функція lm(). (Рис. 3.9).

Код:

*lm.fit=lm(PE ~ AT, data = power)*

*summary(lm.fit)*

```
> summary(lm.fit)

Call:
lm(formula = PE ~ AT, data = power)

Residuals:
    Min      1Q  Median      3Q     Max
-45.951  -3.644   0.101   3.696  23.251

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 497.034120   0.156434  3177.3   <2e-16 ***
AT           -2.171320   0.007443  -291.7   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.426 on 9566 degrees of freedom
Multiple R-squared:  0.8989,    Adjusted R-squared:  0.8989
F-statistic: 8.51e+04 on 1 and 9566 DF,  p-value: < 2.2e-16

>
```

Рис. 3.9.Проста регресійна модель

Отримуємо доступ до оцінок коефіцієнтів (Рис. 3.10).

Код:

*coef(lm.fit)*

*confint(lm.fit)*

```
> coef(lm.fit)
(Intercept)          AT
  497.03412    -2.17132
> confint(lm.fit)
                2.5 %    97.5 %
(Intercept) 496.72748 497.34076
AT           -2.18591  -2.15673
> |
```

Рис. 3.10. Коефіцієнти

Наступний крок – обрахування інтервалів довіри та інтервалів передбачень (Рис. 3.11).

Код:

*predict(lm.fit, data.frame(AT=(c(5, 25, 35))), interval = "confidence")*
*predict(lm.fit, data.frame(AT=(c(5, 25, 35))), interval = "prediction")*

```
> predict(lm.fit, data.frame(AT=(c(5, 20, 35))), interval = "confidence")
       fit      lwr      upr
1 486.1775 485.9377 486.4174
2 453.6077 453.4989 453.7166
3 421.0379 420.7890 421.2869
> predict(lm.fit, data.frame(AT=(c(5, 20, 35))), interval = "prediction")
       fit      lwr      upr
1 486.1775 475.5394 496.8157
2 453.6077 442.9717 464.2437
3 421.0379 410.3996 431.6763
>
```

Рис. 3.11.Обрахування інтервалів довіри та передбачень

Згідно результатів можна побачити, що:

1) 95% -ний довірчий інтервал, пов'язаний із значенням AT = 5, складається (485.9, 486.4), 95% -ний інтервал прогнозування - (475.5, 496.8), при прогнозованій PE = 486.2.

2) 95% -ний довірчий інтервал, пов'язаний із значенням AP = 20, складається (453.5, 453.7), 95% -ний інтервал прогнозування - (442.97, 464.24), при прогнозованій PE = 453.6.
3) 95% -ний довідковий інтервал, пов'язаний із значенням AP = 35, складається (420.8, 421.3), 95% -ний інтервал прогнозу - (410.4, 431.7 ), при прогнозованій PE = 421.04.

Будуємо графік (Рис. 3.12):

Код:

```
abline(lm.fit, lwd = 1, col = "red")
```

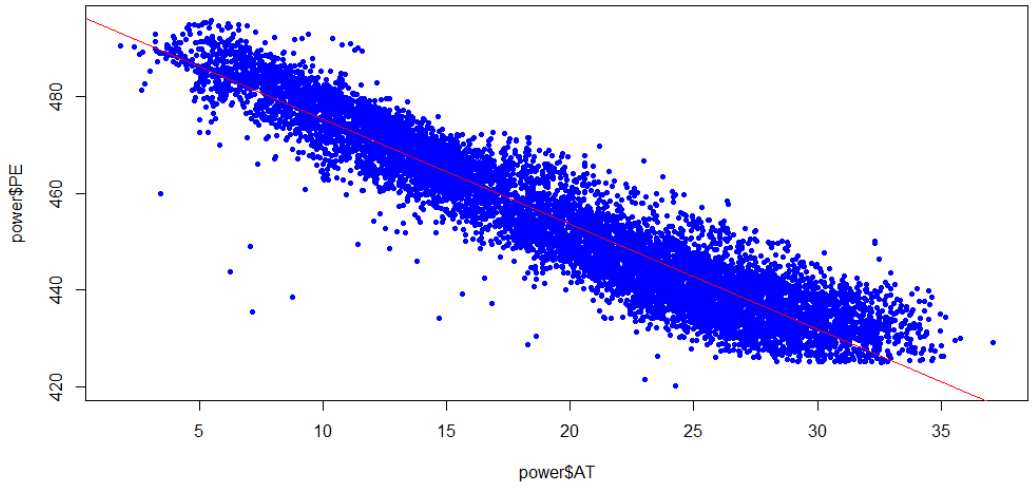



Рис. 3.12Результат прогнозування

Виводимо усі параметри регресійної моделі у вигляді графіків.

Код:

```
par(mfrow=c(2, 2))
plot(lm.fit)
```

Для обрахунку залишків регресійної моделі (Рис. 3.13) використовуються функції residuals() та rstudent().

Код:

```
plot(predict(lm.fit), rstudent(lm.fit))
plot(predict(lm.fit), residuals(lm.fit))
```

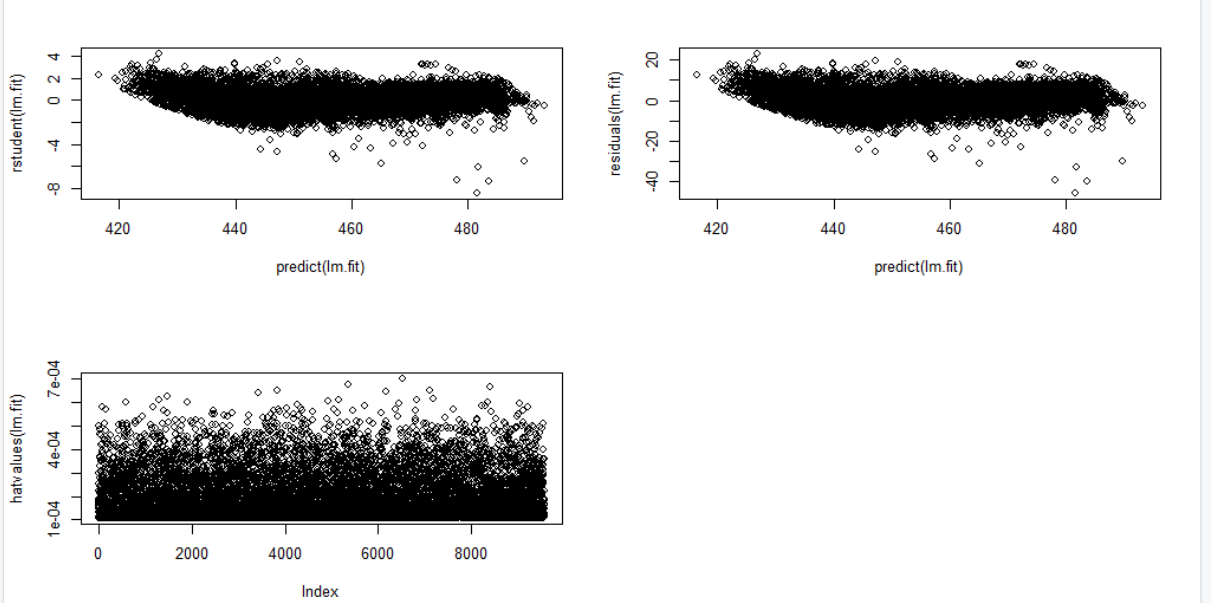



Рис.3.13.Залишки моделі

Досліджуємо функцію на незбалансованість.

Код:

which.max(hatvalues(lm.fit))

В результаті виконання коду можна побачити індекс незбалансованості 6515.

Для оцінки якості прогнозування використаємо метрики оцінки якостіпрогнозування, такі як: MSE, RMSE, MAE, MAPE, та створимо таблицю порівняння якості прогнозування на навчальних даних та тестових (Табл. 3.1).

Таблиця 3.1

Оцінка якості прогнозування

| | Навчальна | Тестова |
|---|---|---|
| $\boldsymbol{R^2}$ | 0.9 | 0.894 |
| MSE | 29.1721 | 30.4757 |
| RMSE | 5.4011 | 5.5205 |
| MAE | 4.2694 | 4.3565 |
| MAPE(%) | 0.9442 | 0.9652 |

Проаналізувавши таблицю можна зробити висновок, що якість прогнозування на тестовому наборі даних гірша ніж на тренувальному.

### 3.4 Множинна лінійна регресія

Для створення множинної лінійної регресії за методом найменших квадратів використовуємо функцію lm(). Створюємо модель для 1 парметру PE (Рис. 3.14).

Код:

*lm.fit=lm(PE~., data=power)*

*summary(lm.fit)*

```
> lm.fit=lm(PE ~ . , data = power)
> summary(lm.fit)

Call:
lm(formula = PE ~ ., data = power)

Residuals:
    Min      1Q  Median      3Q     Max
-43.435  -3.166  -0.118   3.201  17.778

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) 454.609274   9.748512   46.634  < 2e-16 ***
AT           -1.977513   0.015289 -129.342  < 2e-16 ***
V            -0.233916   0.007282  -32.122  < 2e-16 ***
AP            0.062083   0.009458    6.564 5.51e-11 ***
RH           -0.158054   0.004168  -37.918  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.558 on 9563 degrees of freedom
Multiple R-squared:  0.9287,    Adjusted R-squared:  0.9287
F-statistic: 3.114e+04 on 4 and 9563 DF,  p-value: < 2.2e-16

> |
```

Рис. 3.14. Множинна лінійна регресія

Спираючись на p-значення, ми можемо сказати, що між відгуком та усіма предикторами існує сильний взаємозв’язок.

Побудуємо дві моделі: з одним та і у составі з нелінійним предиктором (Рис. 3.15 – Рис. 3.16).

```
> lm.fit1 = lm(PE ~ AT, data = power)
> summary(lm.fit1)

Call:
lm(formula = PE ~ AT, data = power)

Residuals:
    Min      1Q  Median      3Q     Max
-45.951  -3.644   0.101   3.696  23.251

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 497.034120   0.156434  3177.3   <2e-16 ***
AT           -2.171320   0.007443  -291.7   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.426 on 9566 degrees of freedom
Multiple R-squared:  0.8989,    Adjusted R-squared:  0.8989
F-statistic: 8.51e+04 on 1 and 9566 DF,  p-value: < 2.2e-16

>
```

Рис. 3.15. Модель  з одним предиктором

```
> lm.fit2 = lm(PE ~ AT + I(AT ^ 2), data = power)
> summary(lm.fit2)

Call:
lm(formula = PE ~ AT + I(AT^2), data = power)

Residuals:
    Min      1Q  Median      3Q     Max
-48.485  -3.495   0.045   3.565  20.722

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.060e+02  3.409e-01 1484.23   <2e-16 ***
AT          -3.270e+00  3.840e-02  -85.17   <2e-16 ***
I(AT^2)      2.870e-02  9.854e-04   29.12   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.2 on 9565 degrees of freedom
Multiple R-squared:  0.9072,    Adjusted R-squared:  0.9072
F-statistic: 4.674e+04 on 2 and 9565 DF,  p-value: < 2.2e-16

>
```

Рис.3.16Модель із нелінійним предиктором

Ближче до 0 значення p вказує на кращі результати. Але для більш точно порівняння використовуємо функцію anova() (Рис. 3.17).

```
> anova(lm.fit1, lm.fit2)
Analysis of Variance Table

Model 1: PE ~ AT
Model 2: PE ~ AT + I(AT^2)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1   9566 281603
2   9565 258663  1     22939 848.25 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Рис.3.17.Порівняння двух моделей

В даному випадку можна побачити, що значення F  не велике, а значення p далеке від нуля. Тому згідно з цим квадратична модель не дає величезної переваги. Дослідимо модель та побудуємо графіки (Рис. 3.18).

Код:

*par(mfrow=c(2, 2))*

*plot(lm.fit)*

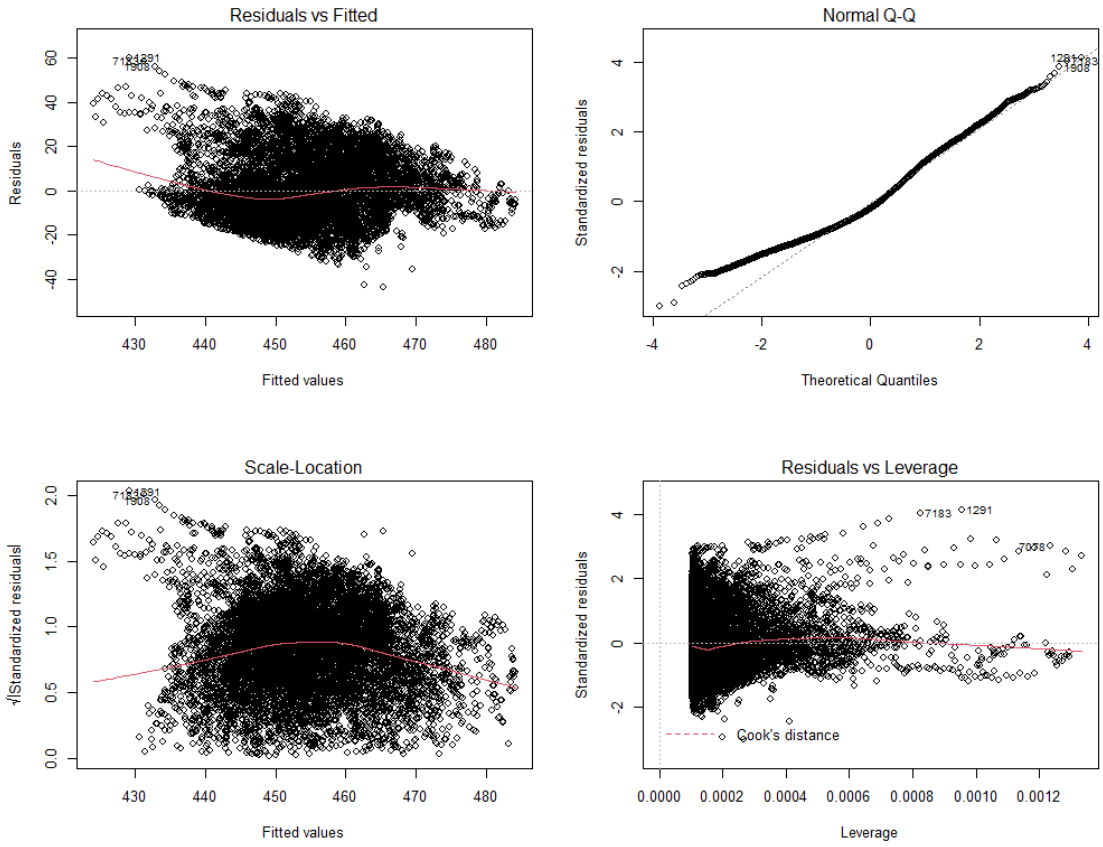



Рис.3.18.  Дослідження моделі

Розширимо модель множинною лінійною регресією для вираження кожної змінної шляхом зміни кожного лінійного компонента $\beta_j x_{ij}$ на нелінійну функцію $f_j(x_{ij})$. Тоді модель даних можливо записати у вигляді:

$$\text{wage} = \beta_0 + f_1(V) + f_2(\text{AT}) + \epsilon;$$

Де: V та AT – кількісні зміні;

Для зображення результату адаптації моделі по методу найменших квадратів (Рис. 3.19).

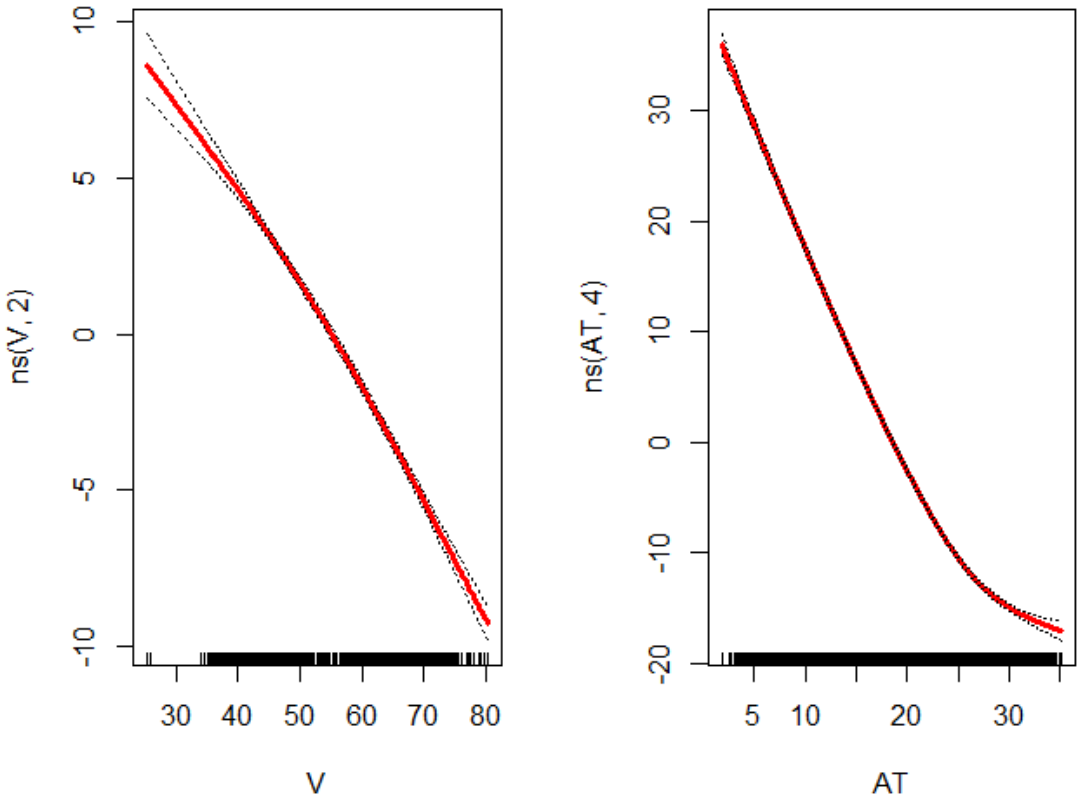



Рис.3.19. Результат виконання

Виконаємо дослідження на знаходження найкращої (Рис. 3.20):

```
> anova(gam, gam2, gam3, test = "F")
Analysis of Variance Table

Model 1: PE ~ ns(V, 2) + ns(AT, 4)
Model 2: PE ~ V + ns(AT, 4)
Model 3: PE ~ ns(AT, 4)
  Res.Df    RSS Df Sum of Sq         F  Pr(>F)
1   6691 137600
2   6692 137760 -1      -160    7.7859 0.00528 **
3   6693 170470 -1    -32710 1590.5849 < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Рис. 3.20Виявлення ліпшої моделі

Дивлячись на результати тесту, можна зробити висновок, що найліпший показник параметру p має модель без предиктора V. Проаналізуємо якість ліпшої моделі використавши метрики якості та занесемо результати до таблиці порівняння (Табл. 3.2).

Таблиця 3.2

Оцінка якості прогнозування

| | Навчальна | Тестова |
|---|---|---|
| $\boldsymbol{R^2}$ | 0.9129 | 0.9081 |
| MSE | 25.4594 | 26.4518 |
| RMSE | 5.0457 | 5.1431 |
| MAE | 3.9508 | 4.0001 |
| MAPE(%) | 0.8722 | 0.8841 |

Проаналізувавши порівняльну таблицю можна побачити, що як і з простою лінійною регресією, прогнозування на тестовому наборі даних демонструє трохи гірший результат. Але якщо порівнювати множинну регресію із простою лінійною, то перша демонструє кращий рівень прогнозування на тестовому наборі даних.

## 3.5 Звичайні регресійні дерева

Перед початком роботи з регресійними деревами потрібно розділити дані на 2 групи: для навчання та для тестування (Рис. 3.21).

```
> trainSize4<-round(nrow(power)*0.8)
> teatSize4<-nrow(power)-trainSize4
> testSize4<-nrow(power)-trainSize4
> trainSize4
[1] 7654
> testSize4
[1] 1914
> set.seed(123)
> training_indices4<-sample(seq_len(nrow(power)), size=trainSize4)
> trainSet4<-power[training_indices4,]
> testSet4<-power[-training_indices4,]
> str(trainSet4)
tibble [7,654 x 5] (S3: tbl_df/tbl/data.frame)
 $ AT: num [1:7654] 10.6 29.9 24.3 11.7 24.8 ...
 $ V : num [1:7654] 41.5 71.3 63.9 41.2 63.6 ...
 $ AP: num [1:7654] 1020 1009 1019 1018 1010 ...
 $ RH: num [1:7654] 91.9 55.7 54 95.7 84.2 ...
 $ PE: num [1:7654] 473 431 420 463 445 ...
> str(testSet4)
tibble [1,914 x 5] (S3: tbl_df/tbl/data.frame)
 $ AT: num [1:1914] 26.27 14.45 7.76 27.23 27.38 ...
 $ V : num [1:1914] 59.4 52.8 42.3 63.9 74.2 ...
 $ AP: num [1:1914] 1012 1024 1009 1014 1010 ...
 $ RH: num [1:1914] 58.8 63.6 83.3 47.2 78.6 ...
 $ PE: num [1:1914] 444 460 484 444 437 ...
>
```

Рис. 3.21. Розділення даних

Створимо графічне зображення дерева (Рис. 3.22):

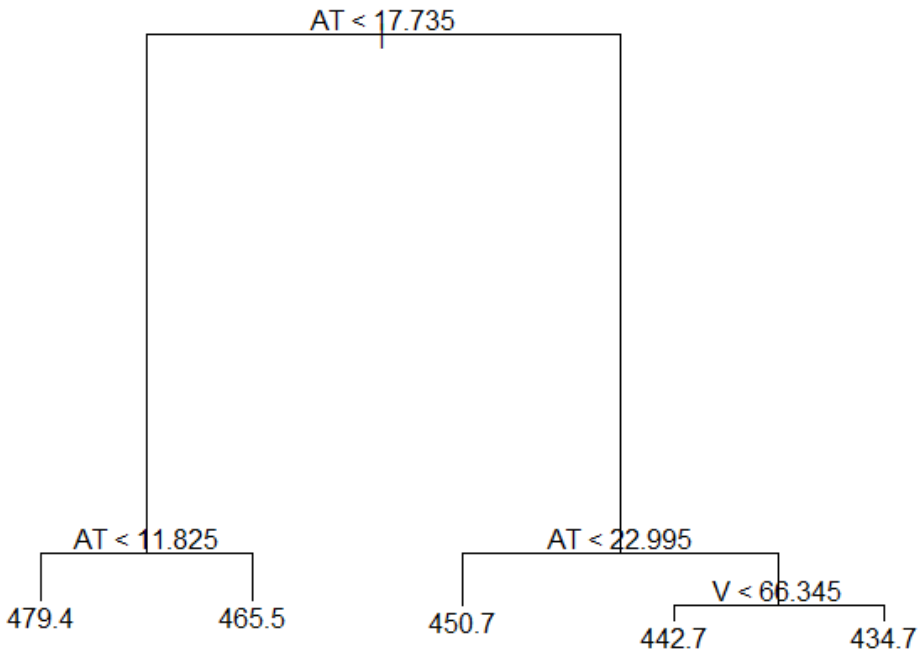



Рис. 3.22. Просте регресійне дерево

Частота помилок на навчальній вибірці дорівнює 33.65. Дерево має 5 кінцевих вузлів. Кінцевий вузол під номером 15 приймає значення 434.7 при V> 66.345, його відхилення дорівнює 39350.

Зробимо відрізання даних для покращення передбачень (Рис. 3.23):

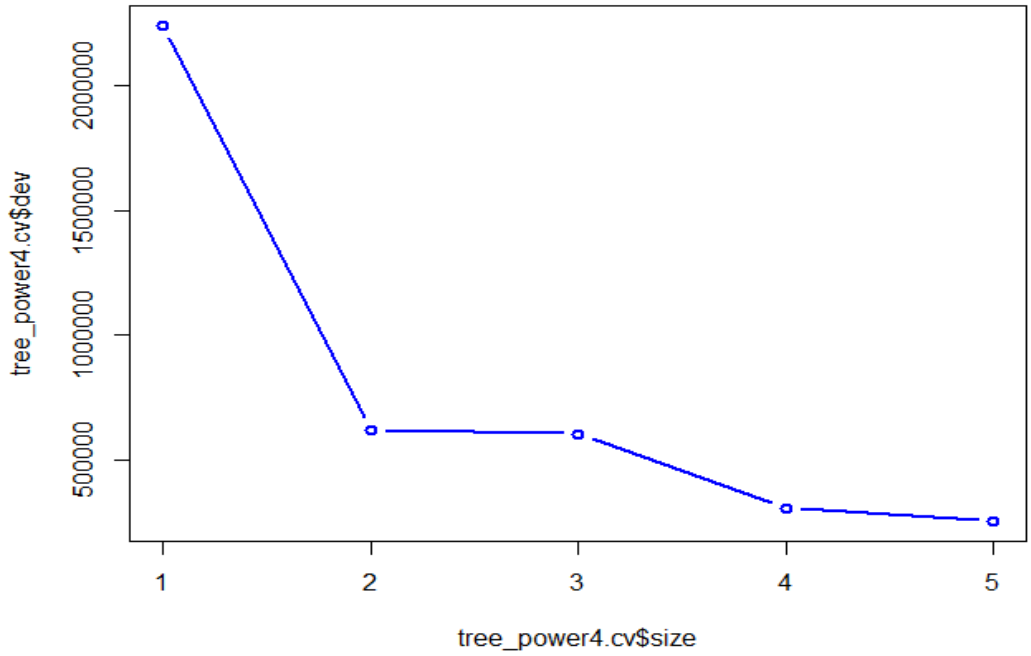



Рис. 3.23.Обрізане дерево

На графіку можна побачити, що оптимальна глиина для дерева – 5, та найменша похибка у вузлі дерева 5.

Оцінимо якість моделі та занесемо значення до порівняльної таблиці (Табл. 3.3).

Табл. 3.3

Оцінка якості прогнозування

| | Навчальна | Тестова |
|---|---|---|
| MSE | 33.6233 | 35.6164 |
| RMSE | 5.7986 | 5.9679 |
| MAE | 4.5099 | 4.6417 |
| MAPE(%) | 0.99 | 1.0194 |

Проаналізувавши порівняльну таблицю можна побачити, що як і раніше, якість прогнозування на тестовому наборі даних гірше ніж на тренувальному. Також провівши порівняння із іншими методами регресії, можна зробити висновок, регресійне дерево має найнижчу якість прогнозування.

### 3.6 Бустинг

Бустинг – алгоритм нагадує беггінг, але дерева будуються послідовно. Кожне наступне дерево зростає з інформації по раніше вирощеним деревам. Також на відміну від побудови одного великого дерева та апроксимації результатів, цей метод використовує повільне навчання (Рис. 3.24).

```
Error: unexpected symbol in "Distribution not"
> boost_power5 = gbm(PE ~ ., data = trainSet5, n.trees = 1000)
Distribution not specified, assuming gaussian ...
> boost_power5
gbm(formula = PE ~ ., data = trainSet5, n.trees = 1000)
A gradient boosted model with gaussian loss function.
1000 iterations were performed.
There were 4 predictors of which 4 had non-zero influence.
> summary(boost_power5)
   var    rel.inf
AT  AT 77.7028007
V    V 20.5617382
AP  AP  0.9401199
RH  RH  0.7953412
```

Рис. 3.24. Створення бустингу

Можна побачити що коефіцієнт важливості найбільший у атрибуту AT – 77.7. Також необхідно побудувати графік впливовості (Рис. 3.25):

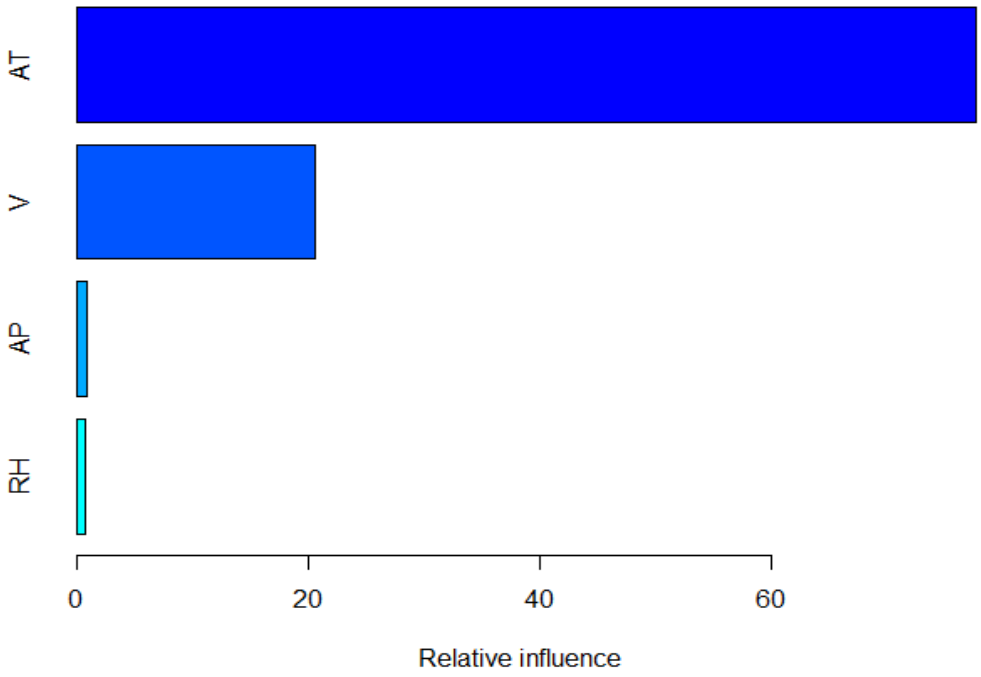



Рис. 3.25 Графік впливовості

Згідно цього графіку впливовості, зміна AT має набагато більший вплив ніж інші змінні. Цей графік відображає впливовість після фіксування інших на постійному рівні.

Побудувавши графік зі значеннями параметра стиснення на осі X і відповідними значеннями MSE на навчальній вибірці на осі Y, можна визначити похибку бустингу (Рис. 3.26).

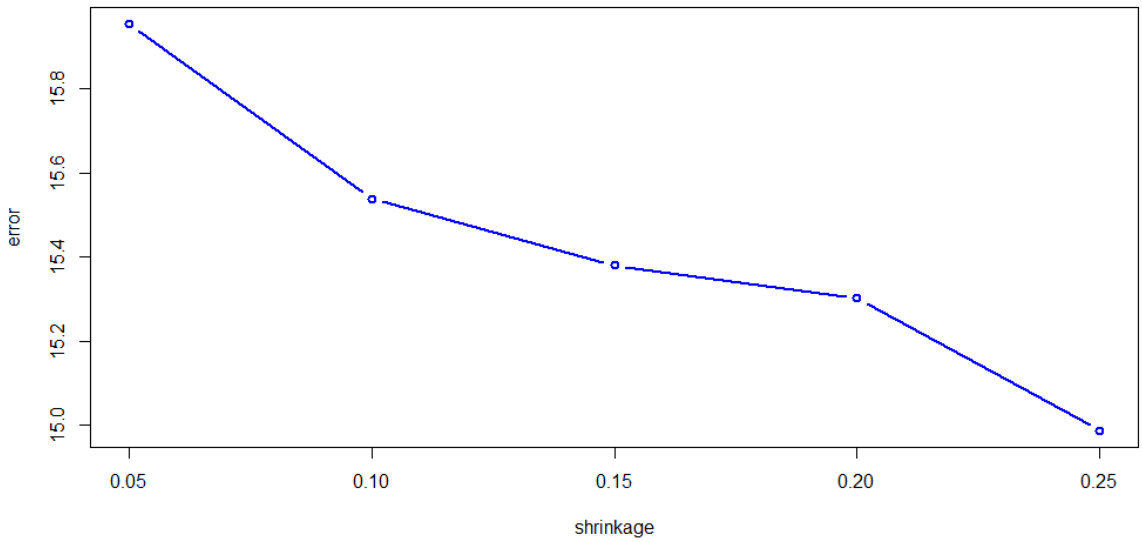



Рис. 3.26. Графік стиснення

Дивлячись на діаграму можна сказати, що найменша похибка при найбільшому значенні стиснення – **15.9**. Далі визначемо якість прогнозування за допомогою інших метрик та занесемо результати до порівняльної талбиці (Табл. 3.4).

Табл. 3.4

Оцінка якості прогнозування

| | Навчальна | Тестова |
|---|---|---|
| MSE | 15.921 | 17.7815 |
| RMSE | 3.9901 | 4.2168 |
| MAE | 3.0904 | 3.1913 |
| MAPE(%) | 0.68 | 0.7019 |

Проаналізувавши таблицю, можна зробити висновок, що використання бустингу призводить до покращення результатів прогнозування. Створення модель демонструє кращі результати ніж інші регресійні моделі.

### 3.7Бегінг та випадковий ліс

Метод бегінга передбачає створення великої кількості копій набору даних, застосовуючи бутстреп, будуючи окремі рішення для кожної з цих копій та подальшої комбінації всіх дерев для створення однієї прогностичної моделі. Важливо відзначити, що кожне дерево побудоване на певному зразку, незалежно від інших дерев.

Застосовуємо методи бегінга та випадкового лісу до набору даних. Нагадаємо, що бегінг є особливим випадком випадкового методу лісу при м = р (Рис. 3.27).

```
> bag_power = randomForest(PE~.,data = trainSet, mtry=4, importance = TRUE)
>
> bag_power

Call:
 randomForest(formula = PE ~ ., data = trainSet, mtry = 4, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 10.81987
                    % Var explained: 96.3
>
```

Рис. 3.27. Застосування бегінгу

При використанні бегінгу із кількістю дерев, яка встановлена за замовченням(n=500), MSEна тестовому наборі – 12.2809 (Рис. 3.28).

```
> p_bag=predict(bag_power, newdata = testSet)
> MSE(p_bag, testSet$PE)
[1] 12.28081
>
```

Рис. 3.28. Результат прогнозування

Для покращення результату необхідно змінити кількість дерев. Провівши тестування на моделях із різною кількістю дерев, а саме: 25, 100, 200, 750, було встановлено, що якість прогнозування найвища саме на моделі, яка містить 500 дерев (Рис. 3.29).

```
> p_bag=predict(bag_power, newdata = testSet)
> MSE(p_bag, testSet$PE)
[1] 12.28081
> bag_power1=randomForest(PE~.,data=trainSet, mtry=4, ntree=25)
> p_bag1=predict(bag_power1, newdata = testSet)
> plot(p_bag1, testSet$PE)
> abline(0,1,col="red")
> MSE(p_bag1, testSet$PE)
[1] 12.74647
> bag_power1=randomForest(PE~.,data=trainSet, mtry=4, ntree=100)
> p_bag1=predict(bag_power1, newdata = testSet)
> plot(p_bag1, testSet$PE)
> abline(0,1,col="red")
> MSE(p_bag1, testSet$PE)
[1] 12.42994
> bag_power1=randomForest(PE~.,data=trainSet, mtry=4, ntree=200)
> p_bag1=predict(bag_power1, newdata = testSet)
> MSE(p_bag1, testSet$PE)
[1] 12.36619
> bag_power1=randomForest(PE~.,data=trainSet, mtry=4, ntree=500)
> p_bag1=predict(bag_power1, newdata = testSet)
> MSE(p_bag1, testSet$PE)
[1] 12.29942
> bag_power1=randomForest(PE~.,data=trainSet, mtry=4, ntree=750)
> p_bag1=predict(bag_power1, newdata = testSet)
> MSE(p_bag1, testSet$PE)
[1] 12.33345
```

Рис. 3.29. Тестування моделей з різною кількістю дерев

Побудуємо графік прогнозування (Рис. 3.30).

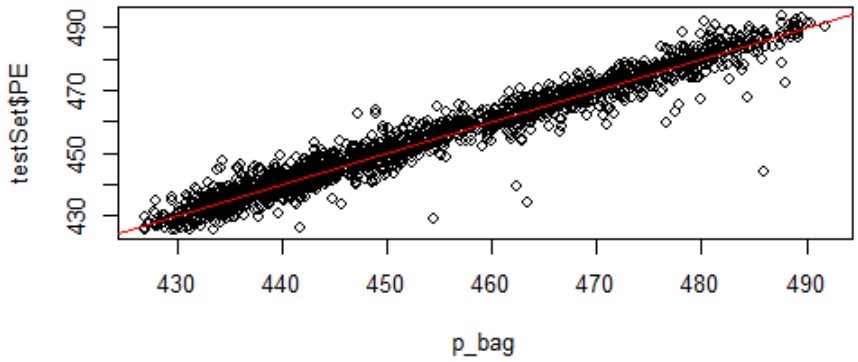



Рис. 3.30. Графік прогнозування

Застосуємо метод випадкового лісу, але змінимо кількість предикторів, які беруть участь у прогнозуванні (Рис. 3.31).

```
> bag_power2 = randomForest(PE~.,data = trainSet, mtry=2, importance = TRUE)
> p_bag2=predict(bag_power2, newdata = testSet)
> MSE(p_bag2, testSet$PE)
[1] 11.6229
> plot(p_bag2, testSet)
```

Рис. 3.31. Модель із двома предикторами

Як можна побачити, модель має MSE– 11.6229 на контрольному наборі даних, що є кращим результатом, ніж використання бегінгу.

Далі визначимо важливість кожної змінної (Рис. 3.32):

```
> importance(bag_power2)
    %IncMSE IncNodePurity
AT 70.13556    1277880.52
V  40.30041     767503.08
AP 49.15196     135144.33
RH 92.94527      45131.58
```

Рис. 3.32 Важливість предикторів

Таблиця демонструє два показники важливості. Перший базується на середньому зменшенні точності прогнозування щодо залишкових даних ("out-of-bag") даних з виключенням відповідної змінної з моделі. Другий показник - це міра середнього збільшення чистоти вузла дерева ("nodepurity") в результаті розділення даних про відповідну змінну. У випадку регресійних дерев чистота вузла виражається за допомогою RSS.

Проаналізувавши таблицю можна побачити що для створення дерев випадкового лісу найбільш важливими змінними є ATта RH.

Далі перевіримо якість прогнозування моделі та занесемо результати до порівняльної таблиці (Табл. 3.5).

Табл. 3.5

Оцінка якості прогнозування

| | Бегінг | | Випадковий ліс | |
|---|---|---|---|---|
| | Тренувальна | Тестова | Тренувальна | Тестова |
| MSE | 2.0921 | 12.2797 | 2.1585 | 11.6229 |
| RMSE | 1.4464 | 3.504238 | 1.4691 | 3.4092 |
| MAE | 1.0111 | 2.418134 | 1.0358 | 2.3563 |
| MAPE | 0.2229 | 0.5334 | 0.2284 | 0.52 |

## Висновки до розділу 3

На цьому етапі роботи було застосовано різні методи регресії для прогнозування вихідної електроенергії електростанції комбінованого типу.

Для кожної моделі було побудовано таблицю визначення точності прогнозування на тренувальному та контрольному наборах даних. Побудувавши загальну порівняльну таблицю та проаналізувавши результати можна зробити висновки щодо точності кожної з моделей на тестовому наборі даних та визначити найкращу модель (Табл. 3.6).

Табл. 3.6

Загальна порівняльна таблиця результатів

| | Проста лінійна регрісія | Множинна регресія | Просте регресійне дерево | Бустинг | Бегінг | Випадковий ліс |
|---|---|---|---|---|---|---|
| MSE | 30.4757 | 26.4518 | 35.6164 | 17.7815 | 12.2797 | 11.6229 |
| RMSE | 5.5205 | 5.1431 | 5.9679 | 4.2168 | 3.504238 | 3.4092 |
| MAE | 4.3565 | 4.0001 | 4.6417 | 3.1913 | 2.418134 | 2.3563 |
| MAPE | 0.9652 | 0.8841 | 1.0194 | 0.7019 | 0.5334 | 0.52 |

Проаналізувавши таблицю порівняння можна побачити що моделі з найкращою якістю прогнозування є методи випадкового лісу та табегінг.

Метод бустингу також виявився досить ефективним в порівняні із методами простої лінійної регресії, множинної регресії та регресійного дерева, але менш ефективним в порівнянні із бегінгом та методом випадкового лісу.

Висока продуктивність випадкового лісу полягає в основному, оскільки нелінійні методи краще включають динаміку даних та перехоплюють нелінійних кореляцій між змінними.

Звичайне регресійне дерево показало найгірший результат в порівнянні із іншими методами регресії. Також можна побачити що множинна регресія виявилася краще у прогнозуванні ніж проста лінійна регресія.

# ВИСНОВКИ

Під час виконання роботи було розроблено, протестовано та оцінено інтелектуальну систему для прогнозування електроенергії електростанції комбінованого типу на основі методів машинного навчання. Було визначено актуальність роботи на сьогоднішній день, поставлена мета, задача, було визначено об’єкт та предмет дослідження.

У першому розділі магістерської кваліфікаційної роботи було розглянуто предметну сферу, визначено поняття машинного навчання. Були розкриті основні методи регресії. У цьому ж розділі було розглянуто аналогічну роботу інших авторів,  було визначено методи регресії, які були використані, та проаналізовано результати оцінки якості моделей та визначено найкращу.

У другому розділі було проаналізовано обрану мову програмування та середовища розробки системи та визначено особливості та переваги в порівнянні із іншими мовами програмування.

Було розглянуто та проаналізовано обраний для розробки системи датасет, який містить 9568 спостережень.

Було визначено та описано усі методи регресії, які використовуються при прогнозуванні. Ознайомлено із метриками оцінки якості прогнозування, такими як, MSE, RMSE, MSE, MAPE.

У третьому розділі було застосовано різні методи регресії для прогнозування вихідної електроенергії електростанції комбінованого типу.

Для кожної моделі було побудовано таблицю визначення точності прогнозування на тренувальному та контрольному наборах даних. Побудувавши загальну порівняльну таблицю та проаналізувавши результати можна зробити висновки щодо точності кожної з моделей на тестовому наборі даних та визначити найкращу модель.

У спеціальній метадичній частині МКР було використано навички з дисципліни «Машинне Навчання» для створення власних лабораторних робіт. Було створено дві лабораторні роботи з методів регресії.

У спеціальній частині з охорони праці та безпеки при надзвичайних ситуаціях було описано приміщення ФОП «Омельчук Едуард Миколаєвич» та розглянуто умови праці у ньому. Виконано розрахунки щодо оснащення приміщення шумоізоляціонними панелями для запобігання зависокого рівня шуму на робочому місці.

Розглянуто забезпечення безпеки персоналу в умовах надзвичайних ситуацій, пов’язаних з пожежною безпекою.

За результатами роботи можна сказати що поставлена задача виконана.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Електронний ресурс]. – Режим доступу: URL:https://coderlessons.com/tutorials/upravlenie/vyuchit-upravlencheskuiu-ekonomiku/metody-regressii
2. [Електронний ресурс]. – Режим доступу: URL:https://bloginnovazione.it/uk/machine-learning/3716/
3. [Електронний ресурс]. – Режим доступу: URL:http://specials.kunsht.com.ua/machinelearning3
4. [Електронний ресурс]. – Режим доступу: URL:http://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
5. [Електронний ресурс]. – Режим доступу: URL:https://www.jmp.com/en_ch/statistics-knowledge-portal/what-is-regression.html
6. [Електронний ресурс]. – Режим доступу: URL:https://corporatefinanceinstitute.com/resources/knowledge/other/multiple-linear-regression/
7. [Електронний ресурс]. – Режим доступу: URL: https://www.solver.com/regression-trees
8. [Електронний ресурс]. – Режим доступу: URL: http://journals.uran.ua/eejet/article/view/245663/246606
9. [Електронний ресурс]. – Режим доступу: URL: https://forecasting.svetunkov.ru/using-r/
10. [Електронний ресурс]. – Режим доступу: URL:https://datascience.stackexchange.com/questions/5345/ide-alternatives-for-r-programming-rstudio-intellij-idea-eclipse-visual-stud
11. [Електронний ресурс]. – Режим доступу: URL: https://libguides.library.kent.edu/statconsulting/r
12. [Електронний ресурс]. – Режим доступу:

URL:https://alexanderdyakonov.files.wordpress.com/2018/10/book_08_metrics_12_blog1.pdf

13. [Електронний ресурс]. – Режим доступу: URL: https://russianblogs.com/article/2973985703/
14. [Електронний ресурс]. – Режим доступу: URL: https://basegroup.ru/community/articles/feature-selection
15. [Електронний ресурс]. – Режим доступу: URL: https://www.machinelearningmastery.ru/linear-and-bayesian-modelling-in-r-predicting-movie-popularity-6c8ef0a44184/
16. [Електронний ресурс]. – Режим доступу: URL: https://russianblogs.com/article/9232898518/
17. [Електронний ресурс]. – Режим доступу: URL:https://www.baslerweb.com/ru/vision-campus/otrasli-i-zadachi/chto-takoe-glubokoe-obuchenie/
18. [Електронний ресурс]. – Режим доступу: URL:https://www.mbureau.ru/blog/osnovnye-ocenki-tochnosti-prognozirovaniya-vremennyh-ryadov
19. [Електронний ресурс]. – Режим доступу: URL: https://habr.com/ru/post/19657/
20. [Електронний ресурс]. – Режим доступу: URL: http://neerc.ifmo.ru/wiki/index.php?title =toggle_view_desktop
21. Combined Cycle Power PlantData Set. Machine learning repository. *UCI Center for Machine Learning and Intelligent Systems* : веб-сайт. URL: http://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant (дата звернення: 04.03.2021).
22. The R Language for Statistical Computing *The R Project* : веб-сайт. URL: https://www.r-project.org/ (дата звернення: 30.04.2021).
23. Траск Э. Грокаем глубокое обучение: навчальний посібник / пер. с англ. А. Кисилев. СПб, 2019, 352 с.

24. Глибовець М.М. Машинне навчання. - К.:НУКМА, 2002.
25. Каллан, Роберт Нейронные сети. Краткий справочник. – М.: Издательский дом «Вильямс», 2016. – 288 с.
26. Люгер Дж.Ф. Искусственный интеллект. – М.: Вильямс, 2003.
27. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. – М: Вильямс, 2019, 480 с.
28. Николенко С. Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангельская. – СПб.: Питер. – Серия «Библиотека программиста», 2018. – 480 с.
29. Нікольский Ю.В., Пасічник В.В., Щербина Ю.М. Системи машинного навчання. - Львів 2010.
30. Пратик Джоши. Искусственный интеллект с примерами на Python. – М: Диалектика, 2019. – 448 с.
31. Рассел С., Норвиг П. Искусственный интеллект: современный подход. – М.: Вильямс, 2007.
32. Тарик Рашид Создаем нейронную сеть. – М.: Диалектика, 2018, 272 с.
33. Хайкин С. Нейронные сети – М.: Вильямс, 2006.
34. Wiki-portal *Портал для изучения машинного обучения* : веб-сайт. URL: https://www.machinelearning.ru.
35. Practical Deep Learning for Cloud, Mobile & Edge Real-World - Python – 2019 – 400 с.
36. Coursera *Coding the matrix* курс для поглибленого вивчення математики для data-science спеціалістів.
37. Головні посилання для вивчення машинного навчання. *Outline of machine learning.* https://en.wikipedia.org/wiki/Outline_of_machine_learning.
38. Курс лекцій Воронцова К.В.
39. Machine Learning Course Ng A. Веб посилання: http://ml-class.org.

40. Hastie T., Tibshirani R., Friedman J. The elements of statical learning: Data Mining, Inference and Prediction. Springer. 2009.

# ДОДАТОК А

Лістинг коду.

```
# First 6 rows
head(power)
# Last 6 rows
tail(power)
# Show summary
summary(power)
# Show dataset column values with types
str(power)
```

---

```
install.packages("ISLR")
install.packages("dplyr")
library(dplyr)
library(ISLR)
library(MASS)
library(class)
names(power)
dim(power)
summary(power)
str(power)
pairs(power)
cor(power)
attach(power)
plot(AT)
```

---

```
trainSize<-round(nrow(power01)*0.7)
testSize<-nrow(power01)-trainSize
trainSize
testSize
set.seed(321)
training_indices<-sample(seq_len(nrow(power01)), size=trainSize)
trainSet<-power01[training_indices,]
```

```
testSet<-power01[-training_indices,]

model1 <- ctree(PE~.,data=trainSet)
summary(model1)
model1
plot(model1)
table(predict(model1), trainSet$PE)

testPrediction <- predict(model1, newdata=testSet)
table(testPrediction, testSet$PE)



trainSize2 <- round(nrow(power) * 0.7)
testSize2 <- nrow(power) - trainSize2
trainSize2
testSize2

set.seed(123)
training_indices2 <- sample(seq_len(nrow(power)),size=trainSize2)
trainSet2 <- power[training_indices2, ]
testSet2 <- power[-training_indices2, ]

set.seed(123)
model2 <- randomForest(PE~., data=trainSet2)
summary(model2)
model2
plot(model2)
testPrediction2 <- predict(model2, newdata=testSet2)
table(testPrediction2, testSet2$PE)
```

---

```
install.packages("MASS")
install.packages("ISLR")

library(ISLR)
```

```
library(MASS)

fix(power)

names(power)

lm.fit=lm(PE ~ AP, data = power)
summary(lm.fit)
coef(lm.fit)
confint(lm.fit)
predict(lm.fit, data.frame(AP=(c(1000, 1015, 1030))), interval = "confidence")
predict(lm.fit, data.frame(AP=(c(1000, 1015, 1030))), interval = "prediction")
#plot(power$AP, power$PE, col = "blue", pch = 20)
abline(lm.fit, lwd = 1, col = "red")
cor(power$AP, power$PE)
#plot(predict(lm.fit), rstudent(lm.fit))
#plot(predict(lm.fit), residuals(lm.fit))
#plot(hatvalues(lm.fit))
#which.max(hatvalues(lm.fit))
pred = predict(lm.fit, data.frame(AP=(c(1000, 1015, 1030))), interval = "prediction")
summary(lm.fit)
lm.fit1 = lm(PE ~ AT, data = power)
summary(lm.fit1)
lm.fit2 = lm(PE ~ AT + I(AT ^ 2), data = power)
summary(lm.fit2)
anova(lm.fit1, lm.fit2)
#cor(power)
#plot(power)
lm.fit=lm(PE ~ . , data = power)
summary(lm.fit)
par(mfrow=c(2,2))
plot(predict(lm.fit), rstudent(lm.fit))
plot(predict(lm.fit), residuals(lm.fit))
plot(hatvalues(lm.fit))
```

```
which.max(hatvalues(lm.fit))
```

---

```
install.packages("ISLR")
install.packages("gam")

library(ISLR)
library(gam)

fix(power)

names(power)

summary(power)

str(power)

library(MLmetrics)
library(gam)
library(ISLR)
gam = lm(PE ~ ns(V, 2) + ns(AT, 4), data = trainSet3)
summary(gam)
par(mfrow=c(1,2))
plot.Gam(gam, se = TRUE, col = "red", lwd = 3)
gam2=lm(PE ~ V + ns(AT, 4), data = trainSet3)
gam3=lm(PE ~ ns(AT, 5), data = trainSet3)
anova(gam, gam2, gam3, test = "F")
preds = predict(gam3, newdata=testSet3)
MSE(preds, testSet3$PE)
MAE(preds, testSet3$PE)
MAPE(preds, testSet3$PE)
```

---

```
install.packages("MASS")
install.packages("ISLR")
install.packages("gbm")
```

```
library(gbm)
library(MASS)
library(ISLR)
library(tree)
library(randomForest)

fix(dataset)
names(dataset)
summary(dataset)
str(dataset)

set.seed(1)
train=sample(1:nrow(dataset),nrow(dataset)/2)
nrow(dataset)
tree_power4 = tree(PE ~ ., trainSet4)
summary(tree_power4)
#tree_power4
#plot(tree_power4); text(tree_power4, pretty = 0)
pred4 = predict(tree_power4, newdata = testSet4)
#plot(pred4, testSet4$PE)
abline(0,1)

tree_power4.cv = cv.tree(tree_power4)
#tree_power4.cv
#plot(tree_power4.cv$size, tree_power4.cv$dev, type= "b", col = "blue", lwd = 2)
prune.power4 = prune.tree(tree_power4, 5)
plot(prune.power4); text(prune.power4, pretty = 0)
prune.pred4 = predict(prune.power4, newdata = testSet4)
mse(testSet4$PE, pred4)
rmse(testSet4$PE, pred4)
mse(testSet4$PE, prune.pred4)
rmse(testSet4$PE, prune.pred4)
pred4 = predict(tree_power4, newdata = trainSet4)
```

```
prune.pred4 = predict(prune.power4, newdata = trainSet4)
mse(trainSet4$PE, pred4)
rmse(trainSet4$PE, pred4)
mse(trainSet4$PE, prune.pred4)
rmse(trainSet4$PE, prune.pred4)
```

---

```
trainSize5<-round(nrow(power)*0.7)
testSize5<-nrow(power)-trainSize5
trainSize5
testSize5
set.seed(123)
training_indices5<-sample(seq_len(nrow(power)), size = trainSize5)
trainSet5<-power[training_indices5,]
testSet5<-power[-training_indices5,]
boost_power5_1 = gbm(PE ~ ., data = trainSet5, n.trees = 1000, shrinkage = 0.05)
boost_power5_2 = gbm(PE ~ ., data = trainSet5, n.trees = 1000, shrinkage = 0.1)
boost_power5_3 = gbm(PE ~ ., data = trainSet5, n.trees = 1000, shrinkage = 0.15)
boost_power5_4 = gbm(PE ~ ., data = trainSet5, n.trees = 1000, shrinkage = 0.2)
boost_power5_5 = gbm(PE ~ ., data = trainSet5, n.trees = 1000, shrinkage = 0.25)
pred_power5_1 = predict(boost_power5_1, newdata = trainSet5, n.trees = 1000)
pred_power5_2 = predict(boost_power5_2, newdata = trainSet5, n.trees = 1000)
pred_power5_3 = predict(boost_power5_3, newdata = trainSet5, n.trees = 1000)
pred_power5_4 = predict(boost_power5_4, newdata = trainSet5, n.trees = 1000)
pred_power5_5 = predict(boost_power5_5, newdata = trainSet5, n.trees = 1000)
e1 = mse(trainSet5$PE, pred_power5_1)
e2 = mse(trainSet5$PE, pred_power5_2)
e3 = mse(trainSet5$PE, pred_power5_3)
e4 = mse(trainSet5$PE, pred_power5_4)
e5 = mse(trainSet5$PE, pred_power5_5)
shrinkage = c(0.05, 0.1, 0.15, 0.2, 0.25)
error = c(e1, e2, e3, e4, e5)
plot(shrinkage, error, col = "blue", type = "b", lwd = 2)

pred_power5_1 = predict(boost_power5_1, newdata = testSet5, n.trees = 1000)
```

```
pred_power5_2 = predict(boost_power5_2, newdata = testSet5, n.trees = 1000)
pred_power5_3 = predict(boost_power5_3, newdata = testSet5, n.trees = 1000)
pred_power5_4 = predict(boost_power5_4, newdata = testSet5, n.trees = 1000)
pred_power5_5 = predict(boost_power5_5, newdata = testSet5, n.trees = 1000)
e1 = mse(testSet5$PE, pred_power5_1)
e2 = mse(testSet5$PE, pred_power5_2)
e3 = mse(testSet5$PE, pred_power5_3)
e4 = mse(testSet5$PE, pred_power5_4)
e5 = mse(testSet5$PE, pred_power5_5)
shrinkage = c(0.05, 0.1, 0.15, 0.2, 0.25)
error = c(e1, e2, e3, e4, e5)
plot(shrinkage, error, col = "blue", type = "b", lwd = 2)
e1
```