МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
В.о. завідувача кафедри інтелектуальних інформаційних систем, канд. техн. наук, доц.
_____Є. В. Сіденко
«____»_____202_ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ПРОГНОЗУВАННЯ ПОКАЗНИКІВ РЕКЛАМНИХ ІНТЕРНЕТ-ЗАСТОСУНКІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.21930102

Виконав студент 6-го курсу, групи 601
_____Є. В. Логутов
«16» лютого 2023 р.

Керівник: д-р техн. наук, професор
_____О. П. Гожий
«16» лютого 2023 р.

Миколаїв – 2023

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
В.о. завідувача кафедри інтелектуальних інформаційних систем, канд. техн. наук, доц.
_____ Є. В. Сіденко
« » 20 р.

ЗАВДАННЯ
на магістерську кваліфікаційну роботу
Логутову Єгору В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи «Прогнозування показників рекламних інтернет-застосунків методами машинного навчання».
Керівник роботи Гожий Олександр Петрович, д-р техн. наук, професор.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» листопада 2022 р. № 199
2. Строк подання студентом роботи 16 лютого 2023 р.
3. Вхідні (початкові) дані до роботи: показники ефективності рекламних оголошень інтернет-застосунків. Очікуваний результат: підхід до прогнозування ефективності реклами на основі методів машинного навчання.
4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):
- аналіз сфери цифрового маркетингу;
- огляд існуючих сервісів для прогнозування показників реклами;
- вибір інструментальних засобів та методів для прогнозування;
- реалізація власного підходу до прогнозування на основі методів машинного навчання;
- оцінка і підвищення ефективності методів.
5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: Охорона праці та безпека у надзвичайних ситуаціях.

_____

_____

_____

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	докт. біол. наук., професор Григор'єва Л. І.	
Методична частина	д-р техн. наук, професор Гожий О. П.	

Керівник роботи __д-р техн. наук, професор Гожий О. П._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Логутов Є. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «__07__» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
Виконання магістерської кваліфікаційної роботи

Тема: Прогнозування показників рекламних інтернет-застосунків методами машинного навчання

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР	01.09.2022	20.10.2022	
2	Отримання завдання на виконання МКР	21.10.2022	10.11.2022	
3	Складання календарного плану на період виконання МКР	11.11.2022	15.11.2022	
4	Огляд літератури за темою дослідження	16.11.2022	27.11.2022	
5	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	28.11.2022	18.12.2022	
6	Аналіз предметної області та розробка технічного завдання. Моделювання результатів	19.12.2022	12.01.2023	
7	Опис фахової частини МКР, зокрема аналіз сфери цифрового маркетингу, огляд існуючих сервісів прогнозування показників реклами, вибір засобів та методів для прогнозування, реалізація власного підходу до прогнозування на основі методів машинного навчання, оцінка і підвищення ефективності методів	13.01.2023	25.01.2023	
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2023	02.02.2023	
9	Попередній захист МКР на засіданні комісії кафедри	03.02.2023	03.02.2023	
10	Корегування роботи за результатами попереднього захисту	04.02.2023	06.02.2023	
11	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	07.02.2023	09.02.2023	
12	Подання МКР рецензенту	09.02.2023	10.02.2023	
13	Рецензування МКР	11.02.2023	12.02.2023	
14	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2023	16.02.2023	
15	Захист МКР перед екзаменаційною комісією (ЕК)	22.02.2023	23.02.2023	

Розробив студент _____Логутов Є. В._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи __ д-р техн. наук., професор Гожий О. П. _____
(*наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«14» листопада 2022 р.

АНОТАЦІЯ

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

Логутова Єгора В'ячеславовича

на тему: **"ПРОГНОЗУВАННЯ ПОКАЗНИКІВ РЕКЛАМНИХ ІНТЕРНЕТ-ЗАСТОСУНКІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ"**

Актуальність даної роботи полягає у дослідженні і застосуванні методів машинного навчання у маркетингу, оскільки це значно прискорює аналітику, що дає змогу швидко отримати статистику та відкоригувати параметри націлювання та розміщення реклами, задля отримання найбільшої ефективності реклами.

Об'єктом дослідження є процес прогнозування показників рекламних інтернет-застосунків методами машинного навчання.

Предметом дослідження є методи та інструментальні засоби аналізу і прогнозування показників рекламних інтернет-застосунків .

Метою роботи є підвищення ефективності аналізу і прогнозування показників рекламних інтернет-застосунків методами машинного навчання.

Результатами виконання роботи є дослідження сфери цифрового маркетингу. Вибір регресійного дерева, випадкового лісу, бустінгу та нейронних мереж як найпопулярніших методів для вирішення проблеми прогнозування показників реклами. Реалізація підходу до прогнозування застосовуючи методи машинного навчання, а також оцінка їх ефективності і покращення якості прогнозування шляхом коригування параметрів.

Дана робота складається з п'яти розділів: аналізу предметної сфери, вибору інструментальних засобів та методів для прогнозування, реалізації методів машинного навчання для прогнозування, методичної частини і охорони праці. Загальний обсяг роботи – 118 сторінок. Магістерська кваліфікаційна робота містить один додаток, 112 рисунків, 7 таблиць і посилання на 46 літературних джерел.

Ключові слова: цифровий маркетинг, рекламні оголошення, прогнозування, мова програмування R, регресійне дерево, випадковий ліс, бустінг, нейронні мережі, TensorFlow, Keras API.

ABSTRACT

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

Yehor Lohutov

“PREDICTION OF INDICATORS OF ADVERTISING INTERNET APPLICATIONS USING MACHINE LEARNING METHODS ”

A relevance of this work lies in the research and application of machine learning methods in marketing, as it significantly speeds up analytics, which allows quickly obtain statistics and adjust the parameters of targeting and placement of advertising, in order to obtain the greatest effectiveness of advertising.

An object of the research is the process of forecasting the indicators of advertising internet applications using machine learning methods.

A subject of research is the methods and tools for analyzing and forecasting the indicators of advertising internet applications.

A purpose of the work is to improve the efficiency of analysis and forecasting of the indicators of advertising internet applications using machine learning methods.

The results of the work are research in the field of digital marketing. Selection of regression tree, random forest, boosting and neural networks as the most popular methods to solve the problem of predicting advertising performance. Implementation of an approach to forecasting using machine learning methods, as well as evaluation of their effectiveness and improvement of the quality of forecasting by adjusting parameters.

This work consists of five sections: analysis of the subject area, selection of tools and methods for forecasting, implementation of machine learning methods for forecasting, methodological part and labor protection. The overall scope of the work is 118 pages. Thesis contains 1 application, 112 figures, 7 tables and 46 sources in it.

Key words: digital marketing, advertising, forecasting, R programming language, regression tree, random forest, boosting, neural networks, TensorFlow, Keras API.

ЗМІСТ

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«ПРОГНОЗУВАННЯ ПОКАЗНИКІВ РЕКЛАМНИХ ІНТЕРНЕТ-ЗАСТОСУНКІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ»

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.21930102

Виконав студент 6-го курсу, групи 601
_____Є. В. Логутов
(підпис, ініціали та прізвище)
«___» _____ 202_ р.

Керівник: д-р техн. наук., професор
(наук. ступінь, вчене звання)
_____О. П. Гожий
(підпис, ініціали та прізвище)
«___» _____ 202_ р.

Миколаїв – 2023

ВСТУП

Визначальною особливістю початку двадцять першого століття стало широке впровадження інформаційних технологій у життя людей. Розповсюдження комп'ютерів підштовхнуло прогрес у багатьох сферах людської діяльності.

Еволюція комп'ютерних технологій дозволила зменшити габарити комп'ютерів то розмірів кишені, перетворивши їх у портативні, мобільні пристрої. Саме сучасні мобільні пристрої дістали широке розповсюдження серед суспільства.

Сьогодні неможливо уявити свій день без смартфону, його використовують для спілкування, розваг і навчання. Завдяки цьому бізнес має можливість швидко і у будь-який момент перехопити увагу користувачів та спрямувати її на власні товари або послуги за допомогою інтеграції реклами у застосунки. Все це призвело до виникнення нового каналу цифрового маркетингу.

Розвиток штучного інтелекту і машинного навчання дозволив залучити новітні технології в якості інструментів маркетингу – головним чином для прогнозування показників реклами.

Застосування методів машинного навчання у маркетингу є вкрай актуальним, оскільки значно прискорює аналітику, що дає змогу швидко отримати статистику та відкоригувати параметри націлювання та розміщення реклами, задля отримання найбільшої ефективності.

Тому головною метою роботи є підвищення ефективності прогнозування за рахунок системного використання методів та технологій машинного навчання.

Об'єктом дослідження є процес прогнозування показників рекламних інтернет-застосунків методами машинного навчання. Предметом дослідження є методи та інструментальні засоби аналізу і прогнозування показників рекламних інтернет-застосунків.

Досягнення поставленої мети обумовлює необхідністю вирішення наступних завдань:

– дослідити сферу цифрового маркетингу;

– проаналізувати існуючи сервіси для прогнозування показників реклами;

– здійснити вибір інструментальних засобів та технологій для розробки власного підходу до прогнозування;

– реалізувати прогнозування на основі методів машинного навчання;

– оцінити і підвищити ефективність методів для досягнення найкращих результатів.

1. АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Опис сфери реклами інтернет-застосунків

1.1.1 Аналіз розвитку пристроїв і застосунків

Стрімка еволюція сучасних технологій сприяла появі портативних персональних комп’ютерів, що стали більш компактною версією класичних настільних ПК, які в свою чергу перетворилися в сучасні мобільні пристрої. Виникнення нових пристроїв підштовхнуло розробку нових операційних систем спеціально адаптованих під них, а також зародженню ринку мобільних застосунків, що приносять значні прибутку своїм розробникам.

У 1973 році компанією Motorola був розроблений перший мобільний девайс. Лише через 20 років з’явився перший кишеньковий комп'ютер – Psion 3. Це був перший пристрій побудований на базі операційної системи EPOC та мови програмування OPL, що дозволяла створювати власні додатки. Головним конкурентом Psion став Palm Pilot. Ключовими особливостями девайсу були сенсорний дисплей і можливість створення додатків на C/C++ [1].

Наступну епоху розвитку застосунків пов’язують зі створенням системи Symbian. Symbian був еволюцією платформи EPOC, в результаті спільної роботи Psion, Ericsson, Motorola і Nokia. Результатами співпраці стало те, що до кінця 2009 року більше 250 мільйонів пристроїв працювало на Symbian. Найбільший внесок в розвиток системи зробила Nokia. Symbian S60 під їх керівництвом виріс в платформу, по потужності і наповненості порівнянну з сучасними.

В сучасному світі занепад Nokia і поява Apple iPhone, зумовило протистояння нових платформ: iOS, Android, Windows Phone і BlackBerry OS.

Офіційно система iOS з'явилася в березні 2008 року, проте фактично існувала з початку 2007. Також у 2008 році була випущена бета-версія середовища для розробки додатків – Software Development Kit. Сьогодні в пакет разом зі стандартними інструментами використання фізичних і програмних можливостей пристрою входить XCode і iPhone Simulator.

Розробка Android почалася в 2005 році: тоді Google придбала компанію Android Inc. Над платформою велася робота 2 роки, через 10 місяців після старту продажів iPhone в Google оголосили про запуск мобільної системи Android, створення Open Handset Alliance – альянсу, що займається її підтримкою і розвитком, а також про пакет для розробників Android SDK. Android заснований на ядрі Linux і віртуальній машині Java. Google відразу надав всім бажаючим розробникам практично необмежені можливості для створення додатків – від Android Native Development Kit до OpenGL ES [2].

Windows Mobile існувала на ринку комунікаторів і кишенькових комп'ютерів починаючи з 2000 року, поки в кінці десятиліття популярність сенсорних смартфонів з супутніми операційними системами не поставила Microsoft перед необхідністю створення нових пристроїв. Ідея полягала в тому, щоб взяти від «старшого брата» все краще, прив'язати це до телефонів Nokia, і тим самим привернути розробників до освоєння нової платформи. Цій меті служила Visual Studio Express, що дозволяє створювати як спеціалізовані додатки, так і кросплатформені. Також WP пропонувала розробнику інструменти Windows Bridge, Expression Blend, XNA, Silverlight.

BlackBerry OS – найменш розвинена операційна система. Перша версія була випущена в 2009 році, але лише після п'яти років вона стала по-справжньому стабільної і функціональною. Особливий напрям розвитку системи був зроблений на зручність користування і корпоративну безпеку.

1.1.2 Еволюція маркетингу у поєднанні з застосунками

Виникнення мобільних пристроїв та застосунків призвело до їх широкого розповсюдження серед суспільства. Сьогодні неможливо уявити свій день без використання смартфону.

Завдяки смартфонам та іншим портативним пристроям бізнес може ефективно створювати та розповсюджувати рекламні кампанії. Прибуток будь-

якої компанії зараз залежить від її маркетингової стратегії, і те, як компанія продає свої послуги чи товари, безпосередньо впливає на розміри доходів.

Маркетинг можна розглядати як інструмент, який використовується для створення попиту, покращення репутації і підвищення конкурентоспроможності бізнесу. Якщо декілька десятиліть тому вартість маркетингу становила значних коштів, то зараз розповсюдження реклами за допомогою мобільних пристроїв та застосунків досить невелика.

Мобільний маркетинг – це тип багатоканального цифрового маркетингу, який спрямований на досягнення цільової аудиторії, використовуючи смартфони та інші портативні девайси [3].

За останні десять років все більше людей переходять з ноутбуків або персональних комп’ютерів на використання мобільних телефонів. Смартфони зарекомендували себе як найбільш швидко зростаючий канал цифрового маркетингу і за прогнозами експертів він зростає щорічно на 25%. Тому мобільний маркетинг цілком спрямований на розповсюдження реклами товарів та послуг спрямованих на користувачів смартфонів.

Мобільний маркетинг може включати в себе широкий набір різноманітних рекламних форматів, включаючи маркетинг у соціальних мережах, магазинах мобільних застосунків і інші типи. Мобільний маркетинг найбільш ефективний серед платформ, що базуються на використанні окремих застосунків.

Еволюція мобільного маркетингу розпочалася у 1973 році, коли Мартіном Купером було винайдено базова ідея роботи СМС каналу, тим не менш його знахідка була забута на двадцять років. У Великобританії у 1992 році виникла ідея використання передачі СМС повідомлень як засобу маркетингу [4].

Декілька років потому на ринку з’явився перший мобільний телефон спроможний підключатись до інтернету. Це стало ключової віхою у розвитку мобільного маркетингу. Поступово бізнес почав усвідомлювати ефективність інвестицій у даний вид маркетингу для привертання уваги нових клієнтів.

З 2002 року, бренди і бізнес почав розсилати через СМС різноманітні пропозиції, скидки на товари та інше до їх клієнтів. Ця стратегії працювала відмінно оскільки оператори мобільного зв’язку гарантували надходження повідомлень покупцям.

З розвитком смартфонів і запуском першого iPhone у 2007 році, з’явилося нове покоління мобільних застосунків. Дана подія змістила фокус з СМС маркетингу на використання мобільних застосунків у рекламі. Застосунки стали каналом через який користувачі виходили до мережі, тому багато компаній почали створювати власні застосунки.

Наступне десятиліття, приблизно з 2010 року, стало періодом розвитку 3G і 4G інтернету. Розробники почали фокусуватися на монетизації мобільних застосунків за допомогою інтеграції цифрових покупок до них. Це стало новим етапом розвитку маркетингу, оскільки мобільні застосунки захоплювали все більше користувачів і користувались все більшим попитом.

З 2017 року користувачі смартфонів почали отримувати близько 100 рекламних оголошень через застосунки щодня. Еволюція мобільного маркетингу і застосунків стала найбільш успішною формою цифрового маркетингу.

Мобільна реклама стала єдиним способом привернути увагу майже кожної людини у світі. Станом на сьогодні у світі було випущено більше смартфонів ніж живих людей. За розрахунками аналітиків прогнозована кількість користувачів мобільних пристроїв на 2023 стане 7.33 мільярдів.

У склад мобільного маркетингу включають декілька рекламних форматів, але його можливості виходять за межі простих рекламних повідомлень. Зараз найбільш наочними прикладами є нижче перераховані.

Мобільні застосунки – одна з найбільш популярних платформ мобільного маркетингу. Сьогодні майже кожна компанія має власний застосунок. За допомогою правильної маркетингової стратегії застосунки можуть значно збільшити трафік і утримання користувачів. Наприклад деякі застосунки можуть використовувати GPS навігацію для того, щоб зрозуміти регіон користувачів і на

його основі пропонувати локальні рекламу і спеціальні пропозиції. Іншим прикладом маркетингової стратегії є використання цифрових покупок у застосунку для розблокування нових можливостей чи підписок на новий контент.

Інтернет сайти оптимізовані для мобільних пристроїв – включають в себе інтуїтивний дизайн, контент, що підлаштовується під розмір екрану смартфону.

Пуш повідомлення – застосунки мобільного маркетингу можуть повідомляти користувача про обмежені у часі пропозиції, скидки на товари чи послуги.

Маючи усі вище наведені переваги, мобільний маркетинг є найбільш перспективною сферою розвитку і з часом повністю замінить традиційні способи реклами.

1.1.3 Структура рекламної сфери інтернет-застосунків

Щоденно мільйони користувачів навколо світу використовують смартфони і більшу частину часу вони проводять у мережі. Це призвело до стрімкого збільшення мобільної аудиторії [5].

Для того щоб зрозуміти взаємодію процесів у мобільному маркетингу, необхідно розглянути основні структурні елементи даної сфери (рисунок 1.1).

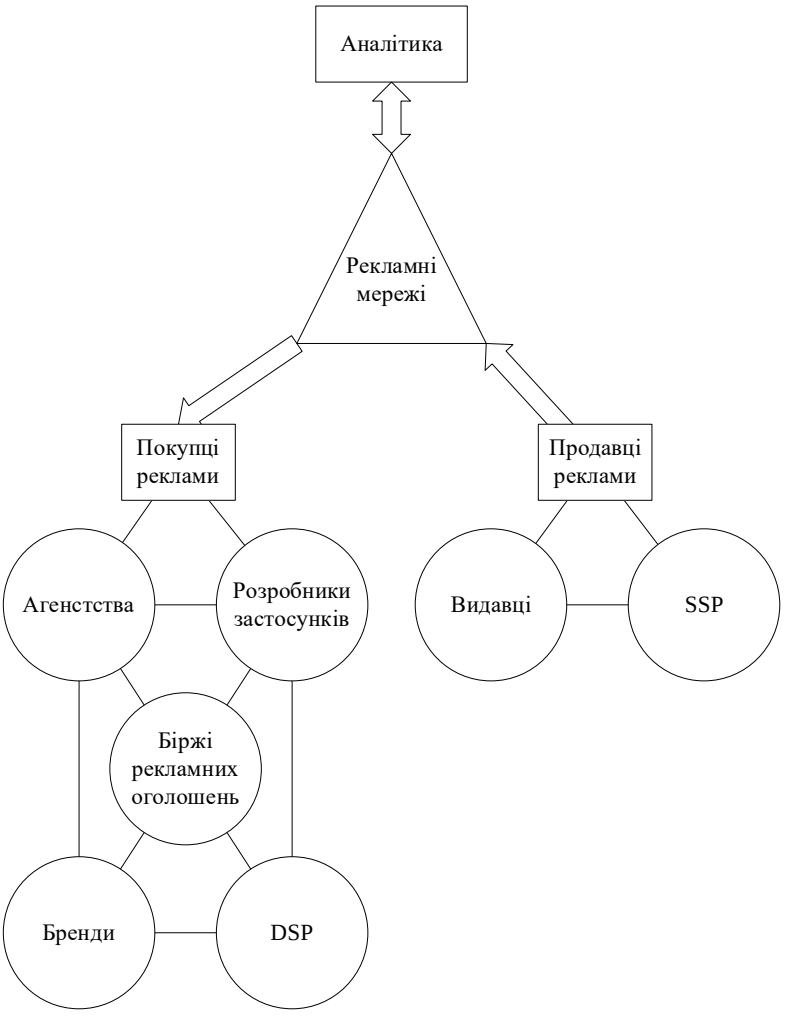

Рисунок 1.1 – Структура рекламної сфери

Екосистему реклами можна розділити на три частини: аналітику, продавців реклами та покупців реклами [6].

Аналітика включає наступне.

Аналітика атрибуції. Незалежні провайдери, що служать фактичними регуляторами екосистем, допомагають маркетологам максимізувати рентабельність інвестицій.

Магазини додатків. Можна розділити на дві категорії, першу самі магазини додатків, що пропонують базову аналітику при встановленні, та другу спеціалізовані компанії, які надають більш досконалі аналітичні інструменти та дані.

Аналітика в додатку. Компанії, які спеціалізуються на аналізі моделей використання додатків, щоб поліпшити його взаємодію з користувачем, а саме рівень утримання його уваги та тривалість взаємодії з додатком.

До продавців реклами відносяться наступні.

Видавці. Видавець – це веб-сайт або застосунок, що продає рекламний ресурс.

SSP. Платформа яка допомагає видавцям автоматично та ефективно управляти продажом своєї реклами багатьом покупцям, щоб максимізувати прибутковість.

Покупці та продавці реклами об’єднує рекламна мережа. Компанія, яка відповідає пропозиціям мобільних веб-сайтів та додатків запитам рекламодавців, є ключовим елементом в галузі.

До покупців реклами належать наступні сутності.

Агентства. Компанії, які керують рекламними кампаніями для кількох рекламодавців.

Розробники застосунків. Компанії, єдиним бізнесом яких є розробка мобільних додатків (Supercell, Shazam, Tinder, Clean Master).

Біржі рекламних оголошень. Автоматизований ринок, який з’єднує рекламодавців та видавців з метою купівлі та продажу рекламних оголошень у реальному часі.

Бренди. Крупні компанії, що постійно присутні у рекламному просторі. Зазвичай вони класифікуються за категоріями, такими як електронна комерція (Macy’s або Amazon), подорожі (American Airlines, Hotels.com), розваги (HBO, Netflix) та CPG (Coca Cola, Kraft).

DSP. Це програмне забезпечення, яке підключається до декількох бірж реклами, автоматично приймаючи складні рішення щодо купівлі медіа, щоб визначити, скільки платити для одного показу оголошення – переважно за допомогою процесу, який називається встановлення ставок у реальному часі.

1.2 Прогнозована реклама

За останні роки штучний інтелект і машинне навчання набули широкого розповсюдження у багатьох сферах, включаючи маркетинг.

Прогнозована реклама (з англ. predictive advertising) – це маркетингове застосування прогнозного аналізу, що в свою чергу є використанням штучного інтелекту, машинного навчання та статистичних алгоритмів для прогнозування показників реклами у майбутньому [7].

У минулому, до винайдення інтернету, маркетологи вже застосовували схожу концепцію для передбачення успішності реклами. З розвитком інформаційних технологій, появою великої кількості даних і статистичних моделей з’явилась можливість їх залучення до процесу створення маркетингових стратегій і купівлі реклами.

Цифровий слід, який залишають користувачі мережі, зростає в геометричній прогресії. Люди постійно ставлять лайки та діляться контентом у соціальних мережах, переглядають веб-сторінки, використовують служби визначення місцезнаходження та надають застосункам спеціальні дозволи для збору даних. Саме ці дані відображають інтереси і потреби користувачів, і якщо їх правильно інтерпретувати, то можна отримати вигоду. З роками збирати дані про користувачів стало простіше, але не зрозуміти, що з ними робити. Здатність вилучити та отримати цінну інформацію з цих даних вимагає спеціального набору навичок, яких немає в багатьох кампаніях.

Для ефективного прогнозування реклами необхідна велика кількість даних. Чим більше даних про аудиторію, там краще і швидше вона зможу створювати точні моделі.

Як і людина, штучний інтелект може спостерігати за поведінкою аудиторії з плином часу та робити прогнози для майбутніх сценаріїв. Єдина відмінність, у порівнянні з людиною, полягає у тому, що штучний інтелект може робити це значно швидше, використовуючи великий набір даних, і з більшою точністю. Наприклад використовувати різноманітні змінні, включаючи поведінку,

демографічні показники, геолокацію, щоб передбачити майбутні показники маркетингу на основі поведінки у минулому.

Застосування методів машинного навчання у маркетингу значно прискорює аналітику, що дає змогу швидко отримати статистику та внести зміни відповідні зміни до націлювання та розміщення реклами, задля отримання найбільшої ефективності.

1.3 Огляд сервісів для прогнозування показників реклами

Відповідно зі зростанням аудиторій і попиту на рекламу з’явилися сервіси, що надають інструменти для аналітики рекламних кампаній будь-яких форматів. Далі розглядаються найбільш популярні з них.

IBM Watson Advertising – набір інструментів для роботи з прогнозованою рекламою від компанії IBM. За допомогою Predictive targeting рекламодавці можуть краще передбачити цільову аудиторію для того, щоб показати правильну рекламу потрібному клієнту у вдалий час. Оголошення, які показуються користувачам, містять більш цілеспрямовану інформацію, що може призвести до збільшення кількості переходів. Accelerator – використовує динамічну креативну оптимізацію, щоб зрозуміти взаємодію користувачів з застосунком у реальному часі та передбачити найкращий рекламний креатив для кожного користувача [8].

Google Ads Forecast metrics – інструмент прогнозування, що враховує рекламну ставку, бюджет, сезонність, якість для оцінки її майбутньої ефективності [9].

Microsoft Advertising Ad Supply Forecasts – сервіс дає змогу прогнозувати покази, витрати та інші показники реклами на основі: критеріїв націлювання, типу кампанії, типу цілі кампанії, налаштувань ставок і витрат, періоду часу. Для створення прогнозів аналізуються вище наведені дані для рекламної кампанії, до уваги також беруться показники ефективності схожих кампаній або рекламодавців і моделюється аукціон рекламних оголошень, процес показу [10].

1.4 Постановка задачі

Дослідивши можливості існуючих сервісів для прогнозування показників реклами, було виявлено декілька значних недоліків.

Першим недоліком є закритість процесів роботи даних сервісів – це означає відсутність можливості перегляду та гнучкої модифікації алгоритмів машинного навчання для досягнення власних цілей. Іншим важливим недоліком є значна вартість щомісячної підписки потрібної для використання даних сервісів.

Враховуючи це, необхідно розглянути та дослідити методи машинного навчання для розробки власного підходу до прогнозування показників реклами застосунків, що дозволить підвищити ефективність маркетингу і скоротити витрати.

Розробка підходу до прогнозування на основі методів машинного навчання дозволяє вирішувати задачі прогнозування за рахунок системного використання різних методів та технологій. Підхід будується на принципах системного підходу та складається з наступних етапів: етап збору даних; етап дослідження та підготовка даних; етап навчання моделі на даних; етап визначення ефективності моделі; етап підвищення ефективності моделі; візуалізація. Головна особливість підходу це наявність процедур оцінювання ефективності моделі та процедур підвищення ефективності моделі за рахунок оцінювання моделей та вибору найбільш ефективних.

Для успішного виконання задачі, її необхідно розбити на проміжні етапи, які виконувати послідовно.

Етап збору даних. Оскільки машини навчаються на певному наборі даних, збір даних є першим етапом. Важливо під час збору даних враховувати їх якість, що проявляється у актуальності, відповідності реальності і відсутності суттєвих помилок. Все це впливатиме на точність створюваної моделі, а помилки зроблені на цьому етапі посилюватимуться з кожним наступним кроком.

Етап підготовки даних. Наступний крок – підготовка даних, що полягає у очищенні набору від непотрібних даних, значень, рядків та стовпців, візуалізації для чіткого розуміння структури і взаємозв'язку між різними змінними. Також розбиття очищених даних на два набори – набір для навчання та набір для тестування.

Етап вибору моделі – підбір алгоритму машинного навчання який відповідає поставленим завданням і набору даних.

Етап навчання моделі – передача підготовлених даних моделі машинного навчання для виявлення закономірностей і прогнозування. Навчання моделі виконується доки вона не зможе виконувати поставлене завдання.

Етап оцінювання прогнозної моделі. Оцінка моделі – перевірка якості навчання, шляхом тестування на раніше невідомих даних. Для цього використовуються раніше відокремленні дані. В результаті отримуються точні вимірювання ефективності, якості та швидкості створеної моделі.

Підвищення ефективності моделі. Після створення і оцінки моделі можна підвищити її точність за допомогою налаштування параметрів. На цьому етапі відбуваються процедури підвищення ефективності моделі (ускладнення, зміна специфікації та топології моделі, використання комбінованих моделей). Це заключний етап підходу результатом, якого є прогностична модель. В результаті отримується модель, яку можливо використовувати для виконання точних прогнозів.

Висновки до розділу 1

В даному розділі було проведено аналіз сфери реклами інтернет-застосунків, а також аналітичних сервісів для прогнозування показників реклами. На основі чого, виявлено недоліки існуючих сервісів та можливість розроблення власного підходу до прогнозування на основі методів машинного навчання.

Досліджено та сформовано основні етапи роботи для досягнення поставленої мети, а саме:

1) етап збору даних;

2) етап дослідження та підготовки даних;

3) етап навчання моделі на даних;

4) етап визначення ефективності моделі;

5) етап підвищення ефективності моделі;

6) візуалізація.

Головною особливість підходу до прогнозування є наявність процедур оцінювання ефективності моделі та процедур підвищення ефективності моделі за рахунок оцінювання моделей та вибору найбільш ефективних.

2. АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА МЕТОДІВ ДЛЯ ПРОГНОЗУВАННЯ

2.1 Середа розробки R

Обробка даних з різноманітних джерел є одним із природніх інструментів людини для пізнання світу. Сьогодні сутність статистичного аналізу полягає в інтерактивному процесі, який включає в себе дослідження, візуалізацію і інтерпретацію потоків даних.

З розвитком інформаційних технологій, появою глобальної мережі та розповсюдженням персональних комп'ютерів значно зросли обсяги даних. Великі компанії такі як Google, Meta, Amazon оперують терабайтами даних клієнтів, а приватні, урядові, академічні інститути зберігають великий обсяг архівних даних і матеріалів. Збільшення обсягу інформації призвело до розвитку наук присвячених аналізу даних. Пів століття тому аналіз даних проводився в обчислювальних центрах, де обчислення з використанням тодішніх комп'ютерів займало значний час, а тому було досить дорогим. Вчені та аналітики заздалегідь і ретельно встановлювали всі параметри аналізу, а також повинні були ретельно переглядати усі результати, залишаючи потрібне і відкидаючи зайве, задля оптимізації часу і скорочення витрат.

Поява сучасних потужних комп'ютерів призвела до прискорення і здешевлення обчислень. Замість попередньої конфігурації усіх параметрів аналізу, робота стала значно інтерактивною, де на будь-якому етапі аналізу можна здійснити трансформацію даних, вставку нових значень, додавання або видалення змінних, без зупинки процесу аналізу. В результаті аналітик має можливість завершити процес аналізу у будь-який момент коли впевниться, що вирішив усі завдання.

Сьогодні існує багато програмних засобів для аналізу даних наприклад: Apache Spark, Matlab, SAS, Microsoft Power BI. Більшість з них є комерційними

обчислювальними системами зі значної ціною підписок. Найбільш популярною, відкритою і безкоштовною альтернативою є середовище R.

R є мовою програмування й середовищем для статистичних обчислень і графічного аналізу, з відкритим кодом, який підтримується великою спільнотою у всьому світі [11]. Головними перевагами є:

1) безкоштовне використання;

2) наявність потужних можливостей візуалізації складних даних;

3) можливість імпортування даних з різних джерел;

4) платформа, що дозволяє писати власні програми;

5) сучасний графічний інтерфейс.

Середа R складається з трьох основних частин. Першою є мова програмування високого рівня R, що дозволяє одним рядком реалізувати різні операції з об’єктами, векторами, матрицями, списками та іншим. Другою частиною є великий набір функцій обробки даних, зібраних в окремі пакети. Третя – розвинена система підтримки, що включає оновлення компонентів, інтерактивну допомогу й освітні ресурси.

Історія створення R розпочинається у 1993 року, коли двоє новозеландських вчених Росс Ихак (Ross Ihaka) і Роберт Джентльмен (Robert Gentleman) анонсували свою нову розробку, яку назвали R. Вони взяли за основу мову програмування розвиненої комерційної системи статистичної обробки даних S-PLUS і створили її безкоштовну відкриту реалізацію, що відрізняється від свого попередника легко розширюваною модульною архітектурою [12]. Незабаром виникла розподілена система зберігання й поширення пакетів до R, відома під абревіатурою “CRAN” (Comprehensive R Archive Network – http://cran.r-project.org), основна ідея організації якої – постійне розширення, колективне тестування й оперативне поширення прикладних засобів обробки даних. Ентузіастами з усього світу станом на січень 2023 року написано 25868 додаткових бібліотек для R, які суттєво розширюють базові можливості системи [13].

Саме з вище наведених фактів випливає те, що R є ідеальним інструментом для виконання даної роботи.

2.2 Програмні пакети середовища R

Досить складно уявити будь-який клас статистичних методів, який ще не реалізований сьогодні в R: лінійні й узагальнені лінійні моделі, нелінійні регресійні моделі, планування експерименту, аналіз часових рядів, класичні параметричні й непараметричні тести, байєсівську статистику, кластерний аналіз та методи згладжування. З допомогою потужних засобів візуалізації, результати аналізу можна узагальнювати у вигляді різноманітних графіків і діаграм. Крім традиційної статистики, розроблений функціонал включає великий набір алгоритмів чисельної математики, методів оптимізації, рішення диференціальних рівнянь, розпізнавання образів та інше.

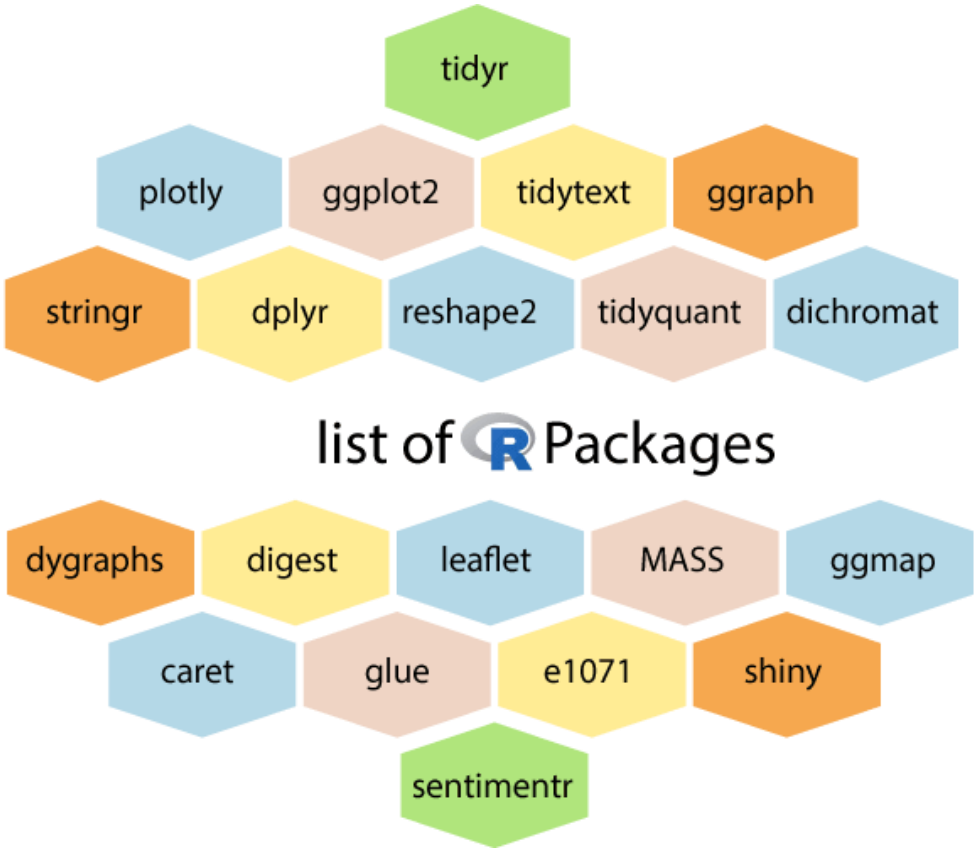

Рисунок 2.1 – Список популярних пакетів R

Далі будуть розглядатися пакети, які будуть застосовуватися під час роботи над проектом.

Пакет corrplot є інструментом візуального дослідження матриці кореляції, який допомагає виявити приховані закономірності серед змінних у наборі даних. Даний пакет має широкий набір налаштування графіків: редагування макету, кольорів, легенди, текстових маркерів. Доступні близько 50 параметрів, однак з них поширені лише деякі, тому для більшості випадків для створення матриці кореляції необхідне викликати лише один рядок коду [14].

MASS – цей пакет містить багато корисних функцій і прикладів даних, зокрема функції для оцінки лінійних моделей за допомогою узагальнених методів найменших квадратів (GLS), підгонки негативних біноміальних лінійних моделей, надійної підгонки лінійних моделей і інше [15].

Бібліотека tree спеціально розроблена для роботи з деревами рішень, дозволяє розробляти, модифікувати і виконувати класифікацію та регресію [16].

Пакет rsample включає в себе функції для створення різних типів вибірок даних і відповідних класів для їх аналізу. Головна мета полягає в тому, щоб надати модульний набір методів, які можна використовувати для: повторної вибірки задля оцінки розподілу статистичних даних і оцінки продуктивності. Сфера застосування пакету лежить у наданні базових будівельних блоків для створення та аналізу вибірок набору даних, але він не містить коду для моделювання [17].

Metrics – реалізація метрик оцінювання в R, які зазвичай використовуються в машинному навчанні. Пакет реалізує метрики для регресії, часових рядів, бінарної класифікації, класифікації і проблеми пошуку інформації. Він не залежить від інших бібліотек і має узгоджений та простий інтерфейс для всіх функцій [18].

Пакет MLmetrics містить набір метрик оцінювання, включаючи функції втрат, корисності, які вимірюють ефективність регресії та класифікації [19].

Бібліотека caret надає функції оптимізації процесу навчання моделі для складних випадків регресії та класифікації. Має декілька функцій для спрощення процесу створення і виконання моделі. Одним з основних інструментів є функція train яку можна використовувати для: виконання моделі і оцінки впливу параметрів налаштувань моделі на якість; вибору оптимальної моделі на основі цих параметрів; оцінки якості моделі на основі навчального набору даних [20].

RandomForest – реалізує алгоритм випадкового лісу Бреймана (на основі Fortran коду Бреймана та Катлера) для задач класифікації та регресії. Його також використовують для оцінки близькості між точками даних [21].

Пакет gbm є реалізацією розширень алгоритмів AdaBoost Фройнда та Шапіра та gradient boosting machine Фрідмана [22].

Головним призначенням пакету reticulate є надання набору інструментів для взаємодії між Python і R, він включає засоби для: виклику Python з R різними способами, включаючи R Markdown, пошук серед Python скриптів, імпорт Python модулів та інтерактивне використання Python у сеансі R; трансляція між об’єктами R і Python (наприклад, між дата фреймами R і Pandas або між матрицями R і масивами NumPy); гнучка прив’язка до різних версій Python, включаючи віртуальні середовища та середовища Conda. Також reticulate вбудовує сеанс роботи Python у сеанс R, забезпечуючи між ними безперебійну, високопродуктивну взаємодію [23].

Пакет tensorflow є інтерфейсом, що сполучає R і відкриту бібліотеку tensorflow.

Пакет keras – інтерфейс до Keras, API нейронної мережі високого рівня.

За допомогою recipes можна створювати конвеєрні послідовності викликів функцій для підготовки набору даних.

Tidyverse – колекція пакетів R розроблених для застосування у data science, де усі компоненти мають спільну філософію, дизайн, граматику і структуру даних [24].

2.3 TensorFlow for R та Keras API

Tensorflow – це бібліотека з відкритим кодом для обчислень і масштабного машинного навчання, що була розроблена Google Brain TensorFlow. Призначена для збору даних, навчання моделей, виконання прогнозів для передбачення майбутніх результатів [25].

Tensorflow об’єднує в собі моделі та алгоритми машинного та глибокого навчання. Використовує Python як зручний інтерфейс і ефективно працює, застосовуючи швидкий і оптимізований C++ код.

У Tensorflow операції для обчислення мають форму графу. Кожен вузол у графі представляє математичну операцію, а кожне з’єднання – дані. Таким чином, замість того, щоб мати справу з низькими деталями, такими як з’ясування правильних способів зв’язати вихідні дані однієї функції з вхідними даними іншої, можна зосередитися на загальній логіці програми [26].

TensorFlow було розроблено у 2015 для внутрішнього використання в Google дослідницькою групою, що займалася вивченням штучного інтелекту і мала назву Google Brain. Сьогодні TensorFlow є однією з найпопулярніших бібліотек, оскільки має відкритий код і широкі можливості для роботи з текстовими даними, у розпізнаванні зображень, голосовому пошуку та багато іншому. Прикладами використання даної бібліотеки сторонніми компаніями є: DeepFace від Meta (система розпізнавання зображень), Siri від Apple (розпізнавання голосу), а також майже кожний сервіс Google [25].

Усі обчислення, пов’язані з TensorFlow, передбачають використання тензорів.

Тензор – це вектор або матриця розмірністю n-вимірів, що представляють типи даних. Значення в тензорі містять ідентичні типи даних із формою, яка є розмірністю матриці. Вектор – одновимірний тензор, а матриця є двовимірним тензором. Скаляр представляє нуль вимірний тензор.

У графі обчислення стають можливими завдяки взаємозв'язкам тензорів. Математичні операції виконуються вузлом тензора, тоді як ребра пояснюють взаємозв'язки введення-виведення між вузлами.

Таким чином, TensorFlow приймає вхідні дані у формі n-вимірного масиву або матриці (відомі як тензори), які проходять через систему кількох операцій і виходять як вихідні дані. Звідси і назва TensorFlow.

В свою чергу Keras є API високого рівня для глибокого навчання, розроблений Google для впровадження нейронних мереж. Він написаний на Python і використовується для полегшення впровадження нейронних мереж, також підтримує серверні обчислення [27].

Оскільки Keras створено з урахуванням високого рівня абстракції від конкретної реалізації нейронних мереж – підтримуються наступні фреймворки: Tensorflow, Teano, PlaidML, MXNet, CNTK (Microsoft Cognitive Toolkit).

З цих п'яти фреймворків розробники TensorFlow інтегрували Keras як офіційний API високого рівня. Keras вбудований в TensorFlow і може використовуватися для швидкого глибокого навчання.

Враховуючи високу популярність і простоту використання, TensorFlow є корисним інструментом, що можна застосувати для прогнозування показників реклами у роботі.

2.4 Оцінка результатів прогнозування

Оцінка якості створеної моделі машинного навчання є важливим етапом проекту. Саме для цього застосовуються різноманітні метрики, які можуть показати досягнутий прогрес, чи навпаки погіршення результатів.

Для задачі регресії використовуються метрики, засновані на обчисленні певної відстані між прогнозованими та фактичним значеннями.

Однією з найпопулярніших метрик є MSE (Mean Squared Error), яка знаходить середнє значення квадрата різниці між фактичним значенням і

значенням прогнозованим регресійною моделлю [28]. Розраховується за формулою 2.1.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \check{y}_j\right)^2, \quad (2.1)$$

де N – кількість входжень у набір даних;

y_j – фактичне значення;

$\check{y}_j$ – прогнозоване значення.

Наступним показником оцінки якості є Mean Absolute Error (MAE) – середнє абсолютне значення різниці між фактичними і прогнозованими значеннями [29]. Розраховується за формулою 2.2.

$$MAE = \frac{1}{N}\sum_{j=1}^{N}\left|y_j - \check{y}_j\right|, \quad (2.2)$$

де N – кількість входжень у набір даних;

y_j – фактичне значення;

$\check{y}_j$ – прогнозоване значення.

Root Mean Squared Error (RMSE) є квадратним коренем з MSE [30]. Розраховується за формулою 2.3.

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(y_j - \check{y}_j\right)^2}, \quad (2.3)$$

де N – кількість входжень у набір даних;

y_j – фактичне значення;

$\check{y}_j$ – прогнозоване значення.

Вище наведені метрики застосовуватимуться для оцінки якості створюваних моделей усіх використаних методів машинного навчання.

Висновки до розділу 2

Результатами роботи в розділі є аналіз та вибір інструментальних засобів та методів для виконання задачі прогнозування показників реклами.

Головним середовищем та мовою програмування обрано R, ключовою перевагою якого є відкритість, безкоштовність і наявність великої кількості пакетів даних, що значно спрощує роботу.

Досліджено наявні пакети R серед яких ключовими є: tree, RandomForest gbm, tensorflow, що безпосередньо відповідатимуть за реалізацію відповідних методів машинного навчання, а також багато допоміжних.

3. РЕАЛІЗАЦІЯ ПІДХОДУ ДО ПРОГНОЗУВАННЯ

3.1 Збір даних

Реалізацію підходу до прогнозування на основі методів машинного навчання розпочинаємо зі збору даних.

Оскільки машини навчаються на певному наборі даних, збір даних є першим етапом. Тому важливо під час збору даних враховувати їх якість, що проявляється у актуальності, відповідності реальності і відсутності суттєвих помилок. Все це впливатиме на точність створюваної моделі, а помилки зроблені на цьому етапі посилюватимуться з кожним наступним кроком.

Джерелом даних є Google AdMob, дані містять показники реклами декількох застосунків. Для завантаження даних у форматі *.csv необхідно було виконати наступні кроки:

1) увійти до аккаунту AdMob за посиланням – https://apps.admob.com;
2) перейти на сторінку звітів, натиснувши на кнопку Reports, що розташовується на бічній панелі веб інтерфейсу;
3) обрати один з звітів за замовчуванням, що стане шаблоном для новоствореного звіту;
4) у шаблоні налаштувати часові межі звіту і обрати усі доступні метрики;
5) натиснути на кнопку SAVE на горі сторінки звіту;
6) вказати ім’я створюваного звіту;
7) натиснути Save, після чого почнеться завантаження даних у вигляді файлу.

На рисунку 3.1 демонструється вміст завантаженого файлу, відкритого за допомогою текстового редактору.

```
1  "","App","Ad.source","Country","Date","Requests","Match.rate.....","Matched.requests","Show.rate.....","Impressions","CTR.....","Clicks","Bid.requests"
2  "1","Coloring1","AdMob Network","Unknown Region","2020-01-01",6075,"4.48%",272,"85.29%",232,"1.72%",4,0,0,0
3  "2","Coloring1","AdMob Network","Andorra","2020-01-01",331,"5.44%",18,"94.44%",17,"0.00%",0,0,0,0
4  "3","Coloring1","AdMob Network","United Arab Emirates","2020-01-01",1243,"36.36%",452,"76.33%",345,"3.19%",11,0,0,0
5  "4","Coloring1","AdMob Network","Afghanistan","2020-01-01",5798,"1.74%",101,"58.42%",59,"0.00%",0,0,0,0
6  "5","Coloring1","AdMob Network","Albania","2020-01-01",7765,"9.67%",751,"81.62%",613,"2.94%",18,0,0,0
7  "6","Coloring1","AdMob Network","Armenia","2020-01-01",5271,"14.59%",769,"79.06%",608,"3.78%",23,0,0,0
8  "7","Coloring1","AdMob Network","Angola","2020-01-01",53,"18.87%",10,"0.00%",0,"",0,0,0,0
9  "8","Coloring1","AdMob Network","Argentina","2020-01-01",72931,"12.52%",9129,"81.27%",7419,"3.91%",290,0,0,0
10 "9","Coloring1","AdMob Network","Austria","2020-01-01",437,"83.98%",367,"69.21%",254,"3.54%",9,0,0,0
11 "10","Coloring1","AdMob Network","Australia","2020-01-01",98,"100.00%",98,"42.86%",42,"4.76%",2,0,0,0
```

Рисунок 3.1 – Вміст завантаженого файлу з набором даних

У таблиці 3.1. послідовно описується структура завантаженого файлу з набором даних.

Таблиця 3.1 – Структура набору даних у файлі формату *.csv

Ім’я стовпця	**Логічне значення**
Відсутнє	Порядковий номер спостереження
App	Назва застосунку
Ad.source	Назва рекламної мережі
Country	Назва країни
Date	Дата спостереження
Requests	Кількість запитів на рекламу
Match.rate.....	Відсоток запитів на рекламу від застосунку, які були заповнені рекламою від мережі
Matched.requests	Загальна кількість запитів заповнених рекламою від мережі
Show.rate.....	Відсоток рекламних оголошень, які були продемонстровані користувачу у застосунку
Impressions	Демонстрації реклами, кількість яких розраховується кожен раз, коли окреме рекламне оголошення відображається у застосунку
CTR.....	Кількість разів, коли користувачі натискаються на рекламу, показану у застосунку, розділене на кількість разів, коли реклама показується у застосунку
Clicks	Кількість разів, коли користувачі натиснули на рекламу
Bid.requests	Загальна кількість запитів-ставок на рекламу в аукціоні
Bids.in.auction	Кількість запитів-ставок, які приймають участь в аукціоні
Winning.bids	Кількість запитів-ставок, що перемогли в аукціоні

3.2 Підготовка даних

Після швидкого перегляду вмісту набору даних, було помічено, що файл містить некоректне форматування, а дані – відсутні значення і потребують подальшої обробки.

Для взаємодії з набором даних використовуються вбудовані інструменти та інтерфейс середовища RStudio. Першим кроком є зчитування набору даних з файлу у оперативну пам'ять (рисунок 3.2), після чого вони відображаються у вікні даних інтерфейсу RStudio (рисунок 3.3).

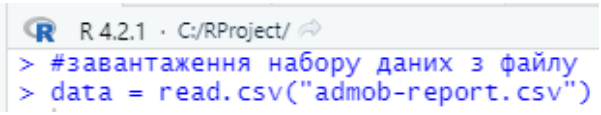

Рисунок 3.2 – Завантаження набору даних з файлу

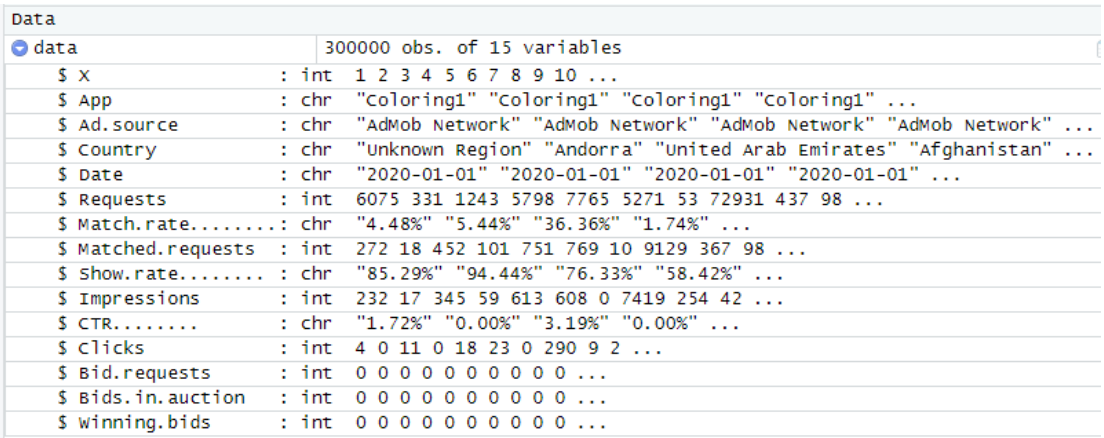

Рисунок 3.3 – Відображення завантажених даних у RStudio

Одразу видаляємо перший стовбець дата фрейму, що містить порядковий номер рядку (рисунок 3.4).

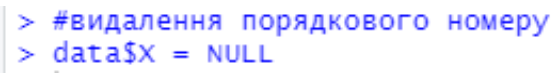

Рисунок 3.4 – Видалення першого стовпця з дата фрейму

Перевірка стовпців Bid.requests, Bids.in.auction, Winning.bids, пов’язаних з аукціоном реклами, за допомогою команди summary() показала, що усі записи

дорівнюють 0. Це означає, що рекламні кампанії у застосунках не використовують механізм аукціону рекламних оголошень, тому вони підлягають видаленню, оскільки не несуть корисної інформації (рисунок 3.5).

```
> #усі значення записів у стовпцях повязаних з аукціоном = 0
> summary(data$Bid.requests)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0
> #видалення стовпця
> data$Bid.requests = NULL
> summary(data$Bids.in.auction)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0
> #видалення стовпця
> data$Bids.in.auction = NULL
> summary(data$Winning.bids)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       0       0       0       0       0
> #видалення стовпця
> data$Winning.bids = NULL
```

Рисунок 3.5 – Видалення даних пов’язаних з аукціоном рекламних оголошень

Для зручності роботи з дата фреймом необхідно перейменувати назви стовпців даних. На рисунку 3.6 наведено назви стовпців до перейменування, безпосередньо команди зміни назв стовпців, а також фінальний результат.

```
> #зміна назв стовпців
> names(data)
 [1] "App"                "Ad.source"          "Country"
 [4] "Date"               "Requests"           "Match.rate........"
 [7] "Matched.requests"   "Show.rate........"  "Impressions"
[10] "CTR........."        "Clicks"
> names(data)[1] = 'app'
> names(data)[2] = 'ad_source'
> names(data)[3] = 'country'
> names(data)[4] = 'date'
> names(data)[5] = 'requests'
> names(data)[6] = 'match_rate'
> names(data)[7] = 'matched_requests'
> names(data)[8] = 'show_rate'
> names(data)[9] = 'impressions'
> names(data)[10] = 'ctr'
> names(data)[11] = 'clicks'
> names(data)
 [1] "app"              "ad_source"        "country"
 [4] "date"             "requests"         "match_rate"
 [7] "matched_requests" "show_rate"        "impressions"
[10] "ctr"              "clicks"
```

Рисунок 3.6 – Перейменування стовпців у дата фреймі

Стовпці date, match_rate, show_rate, ctr є текстовими, хоча містять дати та числові дані, тому необхідне коректне перетворення формату даних для подальшої роботи з ними (рисунок 3.7).

```
> #перетворення тексту у формат дати
> data$date <- as.Date(data$date)
> #перетворення текстових відсотків у числовий формат з застосуванням
 нормалізації
> data$match_rate = as.numeric(sub("%" , "", data$match_rate)) / 100
> data$show_rate = as.numeric(sub("%" , "", data$show_rate)) / 100
> data$ctr = as.numeric(sub("%" , "", data$ctr)) / 100
```

Рисунок 3.7 – Перетворення типів даних стовпців у дата фреймі

Наступним кроком необхідно послідовно перевірити усі дані на коректність і відповідність реальності (рисунок 3.8).

```
> #перевірка коректності даних
> summary(data$date)
        Min.      1st Qu.       Median         Mean      3rd Qu.
"2020-01-01" "2020-03-28" "2020-06-22" "2020-06-26" "2020-09-27"
        Max.
"2020-12-31"
> summary(data$requests)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1      61     343    5380    2087 1539936
> summary(data$match_rate)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.1217  0.4868  0.4826  0.8182  1.0364
> summary(data$matched_requests)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0       8      71    1034     430  281668
> # show_rate містить 25380 NA's
> summary(data$show_rate)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.000   0.614   0.758   0.687   0.852   1.010   25380
```

Рисунок 3.8 – Перевірка даних у date, requests, match_rate, matched_requests, show_rate

Відповідно до значень date, набір даних містить записи починаючи з початку і до кінця 2020 року. Дані у requests, match_rate, matched_requests є коректними оскільки не містять від’ємних значень. Стовпець show_rate містить 25380 відсутніх значень, що необхідно виправити. Для цього застосовано спеціальний алгоритм, що послідовно обходить всі записи з відсутніми значеннями стовпця show_rate і відновлює відповідне значення (рисунок 3.9).

```
> #show_rate NA fix
> #відокремлення NA записів від решти даних
> dataIncorrectShowRate = data[is.na(data$show_rate),]
> data = data[!is.na(data$show_rate),]
> #послідовний обхід всіх записів, що містять NA
> for(i in 1:nrow(dataIncorrectShowRate)) {
+   #витягнення окремого рядку-запису за поточним індексом
+   row = dataIncorrectShowRate[i,]
+   #якщо show_rate = NA то необхідно встановити значення 0 або перерахувати
+   if (!is.na(row$show_rate)) next
+   #show_rate = impressions / matched_requests
+   if (row$impressions == 0 || row$matched_requests == 0)
+     row$show_rate = 0
+   else
+     row$show_rate = row$impressions / row$matched_requests
+   #збереження модифікованого значення
+   dataIncorrectShowRate[i,] = row
+ }
> #включення модифікованих записів до основного дата фрейму
> data = rbind(data, dataIncorrectShowRate)
> #видалення тимчасової змінної
> dataIncorrectShowRate = NULL
```

Рисунок 3.9 – Відновлення значень show_rate

Дані у impressions є цілком коректними, на відміну від ctr, що також містять відсутні значення (рисунок 3.10).

```
> summary(data$impressions)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
     0.0      5.0     52.0    802.7    326.0 244321.0
> # ctr містить 42520 NA's
> summary(data$ctr)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00    0.00    0.04    0.06    0.06    8.00   42520
```

Рисунок 3.10 – Перевірка даних у impressions та ctr

Для вирішення проблеми відсутності значень у ctr, застосовується алгоритм аналогічний до застосованого у випадку відновлення значень show_rate (рисунок 3.11).

```
> #ctr NA fix
> #відокремлення NA записів від решти даних
> dataIncorrectCtr = data[is.na(data$ctr),]
> data = data[!is.na(data$ctr),]
> #послідовний обхід всіх записів, що містять NA
> for(i in 1:nrow(dataIncorrectCtr)) {
+   #витягнення окремого рядку-запису за поточним індексом
+   row = dataIncorrectCtr[i,]
+   #якщо ctr = NA то необхідно встановити значення 0 або перерахувати
+   if (!is.na(row$ctr)) next
+   # ctr = clicks / impressions
+   if (row$clicks == 0 || row$impressions == 0)
+     row$ctr = 0
+   else
+     row$ctr = row$clicks / row$impressions
+   #збереження модифікованого значення
+   dataIncorrectCtr[i,] = row
+ }
> #включення модифікованих записів до основного дата фрейму
> data = rbind(data, dataIncorrectCtr)
> #видалення тимчасової змінної
> dataIncorrectCtr = NULL
```

Рисунок 3.11 – Відновлення значень ctr

Також на рисунку 3.10 можна побачити, що максимальне значення ctr дорівнює 8, що є помилкою оскільки ctr є відсотковим значенням і не може бути більше 1 (100%). Тому необхідно видалити усі записи де значення ctr перевищує 1 (рисунок 3.12).

```
> # Max = 8, CTR не може бути більше 1 (100%) => видаляємо все що більше 1
> summary(data$ctr)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.00000 0.03080 0.04977 0.05710 8.00000
> data = data[data$ctr <= 1,]
```

Рисунок 3.12 – Видалення некоректних записів ctr

Останнім перевірено стовпець clicks, який не містить від'ємних або відсутніх значень, тому не вимагає змін (рисунок 3.13).

```
> summary(data$clicks)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     2.0    35.8    15.0 14206.0
```

Рисунок 3.13 – Перевірка ctr

В результаті, вище описаних дій, отримаємо дата фрейм з 11 змінними і 299799 записами, що повністю відповідає реальності та не містить некоректних записів (рисунок 3.14).

```
data                    299799 obs. of 11 variables
 $ app                : chr  "Coloring1" "Coloring1" "Coloring1" "Coloring1" ...
 $ ad_source          : chr  "AdMob Network" "AdMob Network" "AdMob Network" "AdMob Network" ...
 $ country            : chr  "Unknown Region" "Andorra" "United Arab Emirates" "Afghanistan" ...
 $ date               : Date, format: "2020-01-01" "2020-01-01" ...
 $ requests           : int  6075 331 1243 5798 7765 5271 72931 437 98 53856 ...
 $ match_rate         : num  0.0448 0.0544 0.3636 0.0174 0.0967 ...
 $ matched_requests: int  272 18 452 101 751 769 9129 367 98 3393 ...
 $ show_rate          : num  0.853 0.944 0.763 0.584 0.816 ...
 $ impressions        : int  232 17 345 59 613 608 7419 254 42 2979 ...
 $ ctr                : num  0.0172 0 0.0319 0 0.0294 0.0378 0.0391 0.0354 0.0476 0.0289 ...
 $ clicks             : int  4 0 11 0 18 23 290 9 2 86 ...
```

Рисунок 3.14 – Відображення дата фрейму у RStudio після внесених змін

У таблиці 3.2 наведено опис структури дата фрейму.

Таблиця 3.2 – Структура дата фрейму

Ім’я змінної	**Тип даних**	**Логічне значення**
app	chr	Назва застосунку
ad_source	chr	Назва рекламної мережі
country	chr	Назва країни
date	Date	Дата спостереження
requests	int	Кількість запитів на рекламу
match_rate	num	Відсоток запитів на рекламу від застосунку, які були заповнені рекламою від мережі
matched_requests	int	Загальна кількість запитів заповнених рекламою від мережі
show_rate	num	Відсоток рекламних оголошень, які були продемонстровані користувачу у застосунку
impressions	int	Демонстрації реклами, кількість яких розраховується кожен раз, коли окреме рекламне оголошення відображається у застосунку
ctr	num	Кількість разів, коли користувачі натискаються на рекламу, показану у застосунку, розділене на кількість разів, коли реклама показується у застосунку
clicks	int	Кількість разів, коли користувачі натиснули на рекламу

Оброблені дані повністю готові, тому за допомогою команди наведеної на рисунку. 3.15 вміст дата фрейму зберігаємо у новому файлі для подальшої роботи.

```
#збереження дата фрейму у файл
write.csv(data,"admob-report-modified.csv", row.names = TRUE)
```

Рисунок 3.15 – Збереження дата фрейму у файл

3.3 Дослідження даних

На даному етапі головною метою роботи є дослідження і аналіз структури набору даних, а також відбір необхідних ознак і вивчення взаємозв'язків між змінними.

За допомогою функції str() відображаємо поточну структуру даних (рисунок 3.16).

```
> #відображення структури набору даних
> str(data)
'data.frame':   299799 obs. of  11 variables:
 $ app             : chr  "Coloring1" "Coloring1" "Coloring1" "Coloring1" ...
 $ ad_source       : chr  "AdMob Network" "AdMob Network" "AdMob Network" "AdMob Network" ...
 $ country         : chr  "Unknown Region" "Andorra" "United Arab Emirates" "Afghanistan" ...
 $ date            : Date, format: "2020-01-01" "2020-01-01" "2020-01-01" "2020-01-01" ...
 $ requests        : int  6075 331 1243 5798 7765 5271 72931 437 98 53856 ...
 $ match_rate      : num  0.0448 0.0544 0.3636 0.0174 0.0967 ...
 $ matched_requests: int  272 18 452 101 751 769 9129 367 98 3393 ...
 $ show_rate       : num  0.853 0.944 0.763 0.584 0.816 ...
 $ impressions     : int  232 17 345 59 613 608 7419 254 42 2979 ...
 $ ctr             : num  0.0172 0 0.0319 0 0.0294 0.0378 0.0391 0.0354 0.0476 0.0289 ...
 $ clicks          : int  4 0 11 0 18 23 290 9 2 86 ...
```

Рисунок 3.16 – Структура набору даних

Дата фрейм має незадовільну структуру для регресійної моделі, оскільки містить текстові змінні. Тому необхідно повністю прибрати стовпці app, ad_source, country, date (рисунок 3.17).

```
> #видалення стовпців
> data$app = NULL
> data$ad_source = NULL
> data$country = NULL
> data$date = NULL
```

Рисунок 3.17 – Видалення зайвих змінних з дата фрейму

Залежною змінною у моделі виступає clicks, яка зберігає кількість разів, коли користувачі натиснули на рекламне оголошення. На рисунках 3.18, 3.19 і 3.20 наведено статистику по даній змінній.

```
> summary(data$clicks)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     2.0    35.8    15.0 14206.0
> hist(data$clicks)
> boxplot(data$clicks)
```

Рисунок 3.18 – Зведена статистика змінної clicks

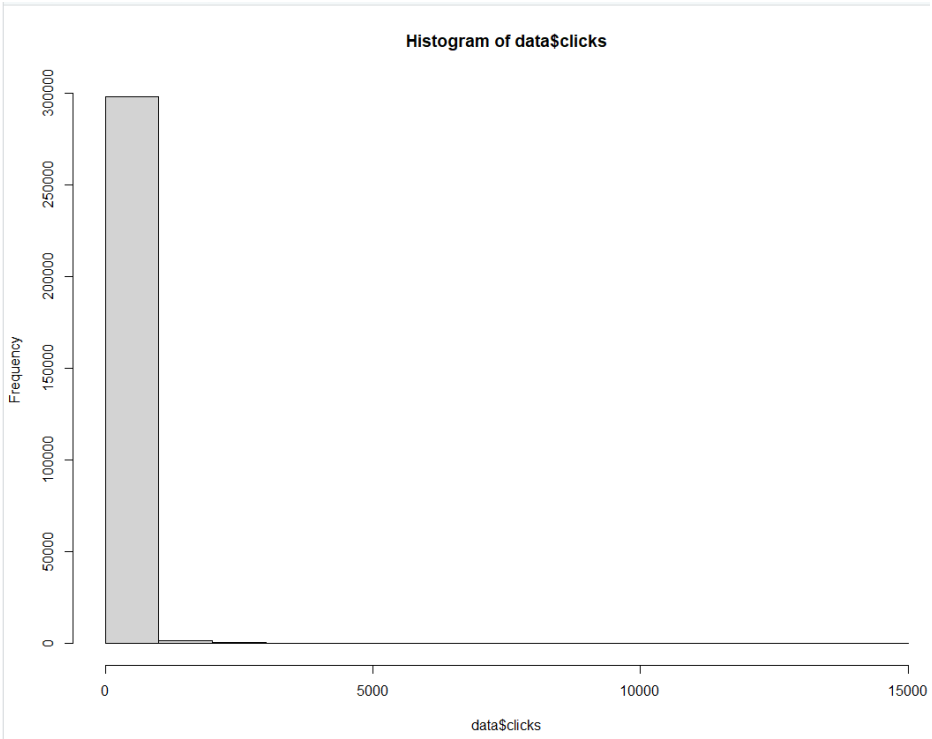

Рисунок 3.19 – Розподіл змінної clicks

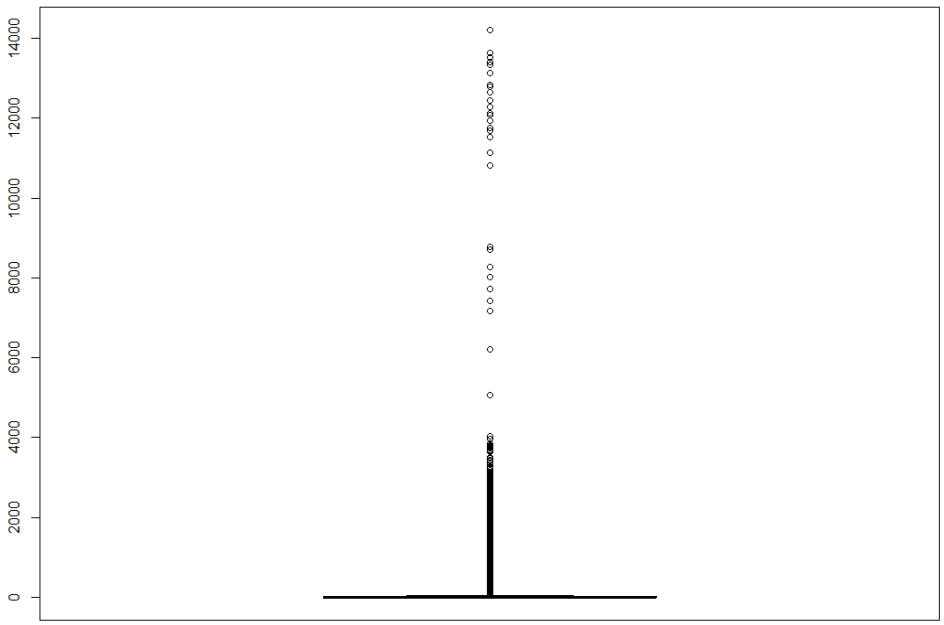

Рисунок 3.19 – Діаграма розмаху змінної clicks

Дивлячись на зведену статистику і розподіл змінної clicks можна стверджувати, що найбільша кількість записів приймає значення менше 2000. Діаграма розмаху вказує на наявність викидів, що буде враховано у подальшому при роботі з даними.

Наступним кроком є визначення взаємозв'язків між незалежними змінними та залежною змінної clicks і між собою. Для цього застосовується розширена матриця кореляції, яка демонструє кореляцію кожної пари змінних з цільового набору даних, а також одночасно з цим візуалізує відносини між числовими об'єктами за допомого діаграм розсіяння для кожної пари змінних (рисунки 3.20, 3.21).

```
> #побудова розширеної матриці кореляції
> pairs = pairs(~requests+match_rate+matched_requests+show_rate+
impressions+ctr+clicks,data=data)
pairs.panels(data)
```

Рисунок 3.20 – Команди побудови розширеної матриці кореляції

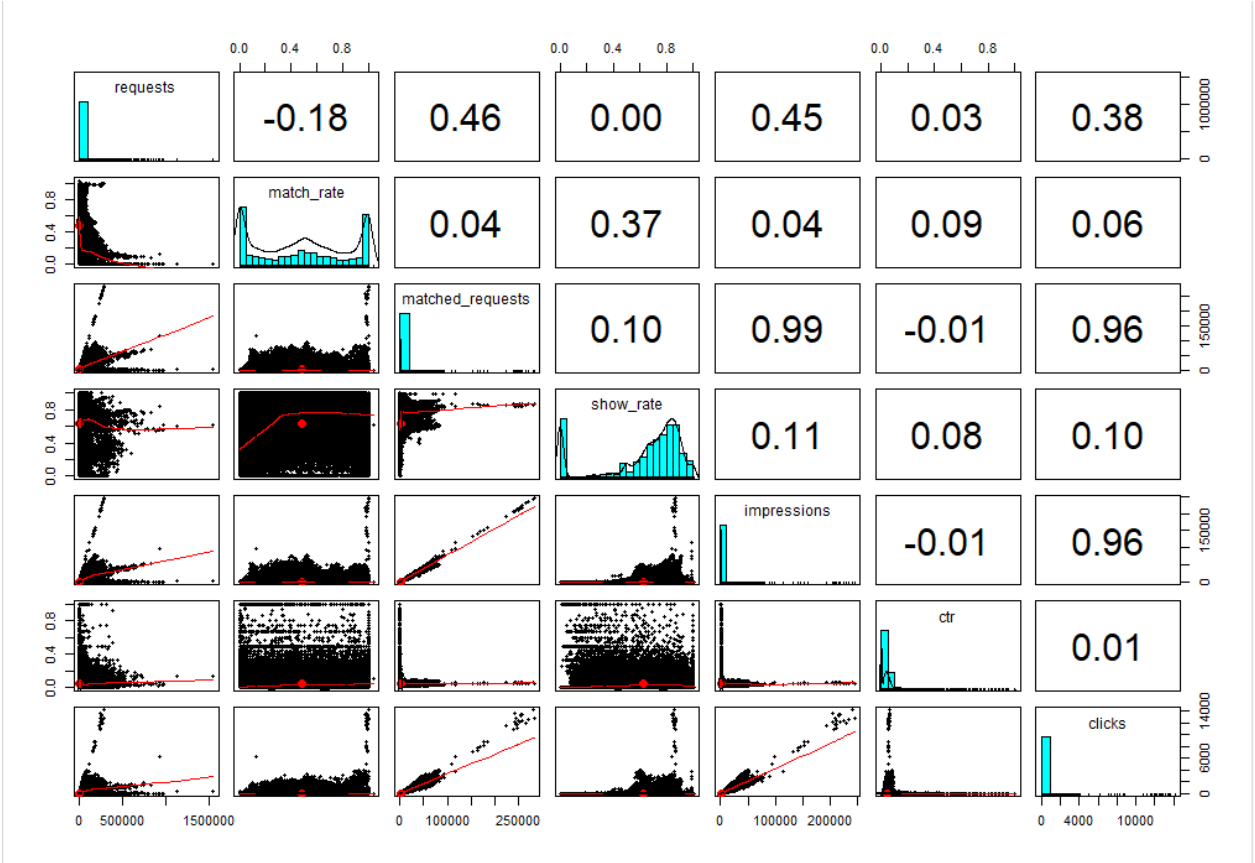

Рисунок 3.21 – Розширена матриця кореляції

За допомогою, вище наведеної, матриці можна побачити, що найбільш сильний зв'язок з залежною змінною clicks мають ознаки requests значення якого становить 0.38, matched_requests значення якого дорівнює 0.96 і impressions – 0.96. Усі інші ознаки мають слабкий зв'язок. Дані показники також підтверджуються матрицями розсіювання для пар відповідних змінних, де за допомогою кривих можна побачити чітку закономірність.

Тому для подальшої роботи з моделями прогнозування для залежної змінної clicks доцільно обрати requests, matched_requests, impressions в якості предикторів.

3.4 Регресійне дерево

Прогнозування clicks у наборі даних розпочинаємо з побудови регресійного дерева. Прості регресійні дерева розбивають набір даних на менші групи, після чого підбирають модель (константу) для кожної підгрупи. Поділ на підгрупи досягається шляхом послідовного бінарного розділу (рекурсивне розділення) на основі різних предикторів. Константа для кожної підгрупи базується на середніх значеннях результуючої змінної для всіх спостережень, які потрапляють у цю підгрупу.

Такий підхід, як правило, є дуже нестабільним і з неточними результатами прогнозування. Однак прості дерева забезпечують фундаментальну основу для більш складних методів на їх основі, таких як випадковий ліс та бутстрап агрегування.

Найвідоміша методологія створення дерев відома як CART (classification and regression tree) розроблена Брейманом та іншими у 1984 році. У даній роботі для побудови дерев застосовується бібліотека tree.

Перш ніж приступити до створення моделі необхідно завантажити набір даних, раніше збережений на етапі підготовки даних, а також видалити зайві змінні (рисунок 3.22).

```
> set.seed(2022)
> options(scipen=999)
> #clicks ~ requests + matched_requests + impressions
> #завантаження набору даних
> data = read.csv("admob-report-modified.csv")
> #видалення зайвих змінних
> data$X = NULL
> data$country = NULL
> data$match_rate = NULL
> data$show_rate = NULL
> data$ctr = NULL
> data$app = NULL
> data$ad_source = NULL
> data$date = NULL
```

Рисунок 3.22 – Завантаження набору даних

Також необхідно прибрати наявні викиди, що були помічені під час дослідження набору, за допомогою фільтрації даних з застосуванням умов наведених на рисунку 3.23.

```
> #видалення викидів даних
> data = data[data$matched_requests < 100000,]
> data = data[data$impressions < 100000,]
```

Рисунок 3.23 – Видалення викидів у наборі даних

Наступним кроком застосуємо нормалізацію методом Min-Max, що дозволить змінити масштаб значень змінних у діапазоні від 0 до 1 (рисунок 3.24).

```
> #нормалізація змінних
> process <- preProcess(as.data.frame(data), method=c("range"))
> data <- predict(process, as.data.frame(data))
```

Рисунок 3.24 – Нормалізація даних

Після чого, за допомогою пакету rsample, розділяємо набір даних на тренувальну і тестову вибірки у пропорції 80% на 20% (рисунок 3.25).

```
> #розподіл набору даних на тестову і тренувальну вибірку
> split <- initial_split(data, 0.8)
> train <- training(split)
> test <- testing(split)
```

Рисунок 3.25 – Створення тренувальної і тестової вибірок

Регресійне дерево створюється за допомогою функції tree() (рисунок 3.26).

```
> tree = tree(clicks ~ requests + matched_requests + impressions, train)
> summary(tree)

Regression tree:
tree(formula = clicks ~ requests + matched_requests + impressions,
    data = train)
Variables actually used in tree construction:
[1] "matched_requests" "impressions"
Number of terminal nodes:  7
Residual mean deviance:  0.0001325 = 31.78 / 239800
Distribution of residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.2485000 -0.0014050 -0.0011570  0.0000000  0.0005794  0.4094000
```

Рисунок 3.26 – Створення регресійного дерева

Відповідно до, виведеної функцією summary(), інформації формула складається з залежної змінної clicks і предикторів requests, matched_requests, impressions, джерелом даних є раніше створена тренувальна вибірка. У побудові дерева були задіяні дві з трьох змінних – matched_requests та impressions. Для кращого розуміння регресійного дерева необхідно графічно зобразити його структуру (рисунки 3.27, 3.28).

```
> plot(tree)
> text(tree, pretty = 0)
```

Рисунок 3.27 – Команди створення візуальної репрезентації структури регресійного дерева

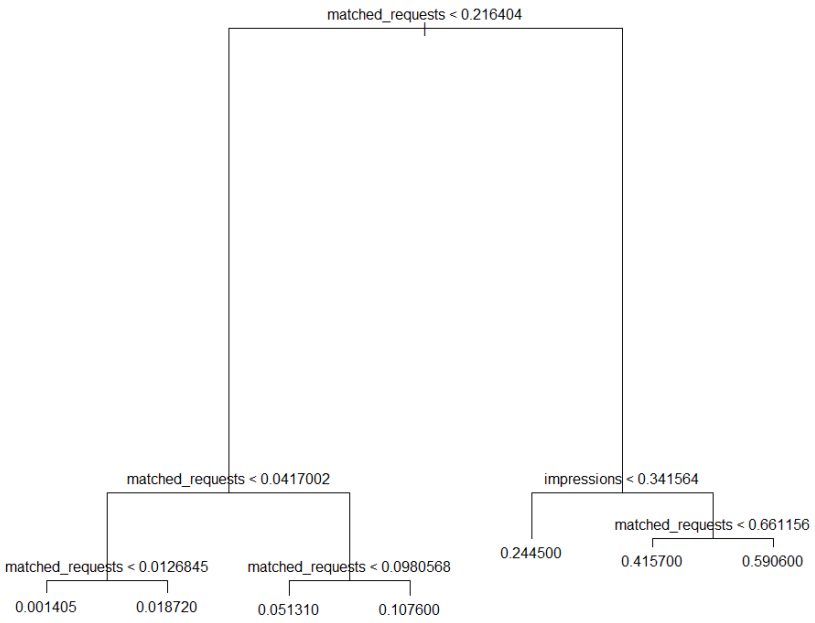

Рисунок 3.28 – Структура регресійного дерева

Розділення кореневого вузла призводить до утворення двох великих гілок, де ліва гілка відповідає умові спостереження matched_requests < 0.216404, а права matched_requests >= 0.216404.

В свою чергу ліва гілка містить внутрішній вузол з умовою matched_requests < 0.0417002, який розділяється на ще два внутрішніх вузла з умовами matched_requests < 0.0126845 і matched_requests < 0.0980568, які остаточно утворюють 4 кінцевих вузла.

Права гілка дерева спрямовує у внутрішній вузол з умовою impressions < 0.341564, який розділяється на один кінцевий вузол і ще один внутрішній вузол з умовою matched_requests < 0.661156, який утворює два кінцевих вузла.

В результаті регресійне дерево складається з кореневого вузла, п’яти внутрішніх вузлів і семи кінцевих вузлів.

Останнім кроком є оцінка якості регресійного дерева з застосуванням тестової вибірки і функції predict(). Змінна y зберігає у собі результати прогнозування clicks на тестовій вибірці, змінні x і x_y необхідні для попарної побудови графіків залежності прогнозованої змінної clicks від змінних matched_requests та impressions.

На рисунках 3.29, 3.30 наведені послідовність команд необхідних для прогнозування з застосуванням тестової вибірки і побудови першого графіка залежності clicks від matched_requests.

```
> x <- test$matched_requests
> y <- predict(tree, test)
> x_y = data.frame(x, y)
> ggplot(train) +
+   geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
+   geom_smooth(data = data.frame(train$matched_requests, train$clicks), mapping = aes(x = train$matched_reques
ts, y = train$clicks, color = "train smoothing")) +
+   geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
+   geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.29 – Застосування predict() і побудова першого графіка

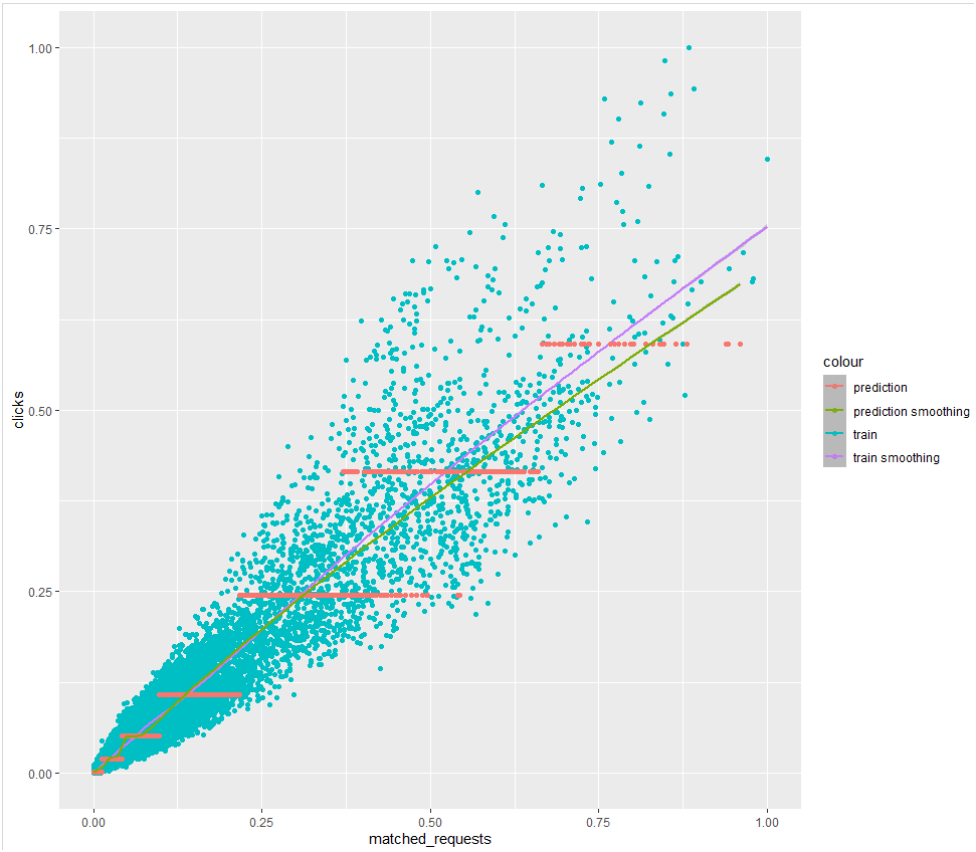

Рисунок 3.30 – Графік залежності значення clicks від matched_requests

Синіми маркерами позначені дані тренувальної вибірки, де вісь ордината представляє значення залежної змінної clicks, а абсциса – matched_requests. Червоні маркери позначають прогнозовані значення тестової вибірки. Лінії демонструють тренд зміни значень для даних тренувальної вибірки (фіолетовий колір) і тестової вибірки (зелений колір).

На рисунках 3.31, 3.32 наведені послідовність команд необхідних для побудови другого графіка залежності clicks від impressions.

```
> x <- test$impressions
> x_y = data.frame(x, y)
> ggplot(train) +
+   geom_point(aes(x = impressions, y = clicks, color = "train")) +
+   geom_smooth(data = data.frame(train$impressions, train$clicks), mapping = aes(x = train$impressions, y = tr
ain$clicks, color = "train smoothing")) +
+   geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
+   geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.31 – Команди побудови другого графіка

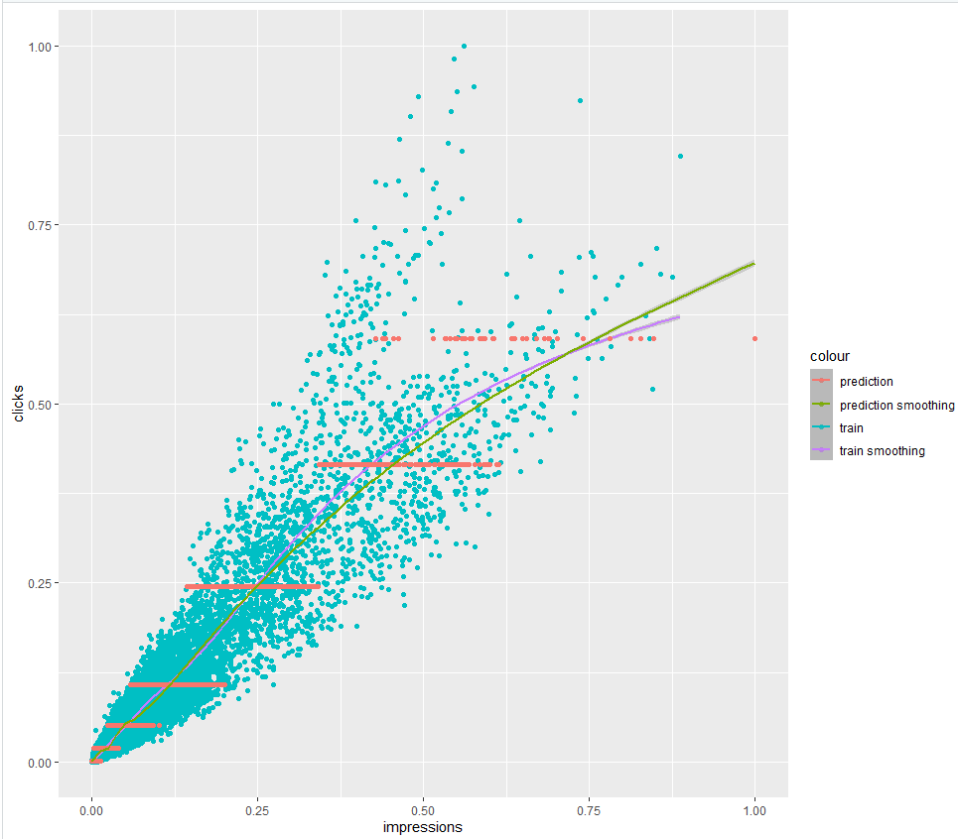

Рисунок 3.32 – Графік залежності значення clicks від impressions

Другий графік має позначення аналогічні до першого окрім того, що вісь абсцис позначає impressions.

Також для оцінки точності прогнозування використані наступні метрики – MSE, MAE, RMSE, значення яких наведено на рисунку 3.33 і у таблиці 3.3.

```
> #calculate MSE
> mean((test$clicks - y)^2)
[1] 2208.425
> #calculate MAE
> mae(test$clicks, y)
[1] 14.64581
> #calculate RMSE
> sqrt(mean((test$clicks - y)^2))
[1] 46.99388
```

Рисунок 3.33 – Розрахунок метрик оцінки точності прогнозування

Таблиця 3.3 – Оцінка точності прогнозування

Модель	**MSE**	**MAE**	**RMSE**
Регресійне дерево	2208.425	14.64581	46.99388

За результатами тестування моделі можна побачити значне відхилення точності прогнозування, порівняно з навчальною вибіркою.

3.5 Випадковий ліс

Наступним застосованим методом прогнозування є випадковий ліс, який базується на використані декількох регресійних дерев на навчальних вибірках створених методом бутстрепу.

Випадковий ліс базується на тих же принципах, що й дерева рішень та бегінг. Об’єднання дерев у групи вводить у даний метод випадковий елемент, який зменшує дисперсію та покращує ефективність прогнозування. Однак дерева не є повністю незалежними одне від одного, оскільки всі предиктори враховуються для кожного дерева.

Для реалізації даного методу використовується бібліотека randomForest, завантаження і підготовка набору даних відбувається ідентичним до регресійного дерева способом (рисунок 3.34).

```
R 4.2.1 · C:/RProject/
> library(randomForest)
> library(rsample)
> set.seed(2022)
> options(scipen=999)
> #завантаження набору даних
> data = read.csv("admob-report-modified.csv")
> #видалення зайвих змінних
> data$X = NULL
> data$country = NULL
> data$match_rate = NULL
> data$show_rate = NULL
> data$ctr = NULL
> data$app = NULL
> data$ad_source = NULL
> data$date = NULL
> #видалення викидів даних
> data = data[data$matched_requests < 100000,]
> data = data[data$impressions < 100000,]
> #нормалізація змінних
> process <- preProcess(as.data.frame(data), method=c("range"))
> data <- predict(process, as.data.frame(data))
```

Рисунок 3.34 – Завантаження набору даних для випадкового лісу

Розбиття набору даних на тренувальну і тестову вибірки відбувається з застосуванням бібліотеки rsample, у пропорції 80% для тренувальної вибірки і відповідно 20% для тестової (рисунок 3.35).

```
> #розподіл набору даних на тестову і тренувальну вибірку
> split <- initial_split(data, 0.8)
> train <- training(split)
> test <- testing(split)
```

Рисунок 3.35 – Створення тренувальної і тестової вибірок

Для виконання випадково лісу застосована функція randomForest(), в яку передаються: залежна змінна, перелік предикторів, джерело даних (навчальна вибірка), параметр, що вказує на кількість предикторів, що розглядаються для створення кожного вузла і кількість створюваних дерев (рисунок 3.36).

```
> bag = randomForest(clicks ~ requests + matched_requests + impressions, data = train,
 mtry = 3, importance = TRUE, ntree = 50)
> bag

Call:
 randomForest(formula = clicks ~ requests + matched_requests +      impressions, data
 = train, mtry = 3, importance = TRUE, ntree = 50)
               Type of random forest: regression
                     Number of trees: 50
No. of variables tried at each split: 3

          Mean of squared residuals: 0.00004449221
                    % Var explained: 96.77
```

Рисунок 3.36 – Створення випадкового лісу

Застосувавши функцію importance(), можна побачити важливість кожної змінної при створенні випадкового лісу (рисунок 3.37).

```
> importance(bag)
                  %IncMSE IncNodePurity
requests         55.83920      12.89087
matched_requests 63.95334     300.19641
impressions      15.20740      17.32550
> plot(bag)
```

Рисунок 3.37 – Важливість змінних

Згідно з рейтингом, найбільш важливим предикторами є matched_requests з показником у 63.95334% і requests з показником 55.83920%.

На рисунку 3.38 зображено графік зменшення похибки зі збільшенням кількості створюваних дерев під час виконання випадкового лісу.

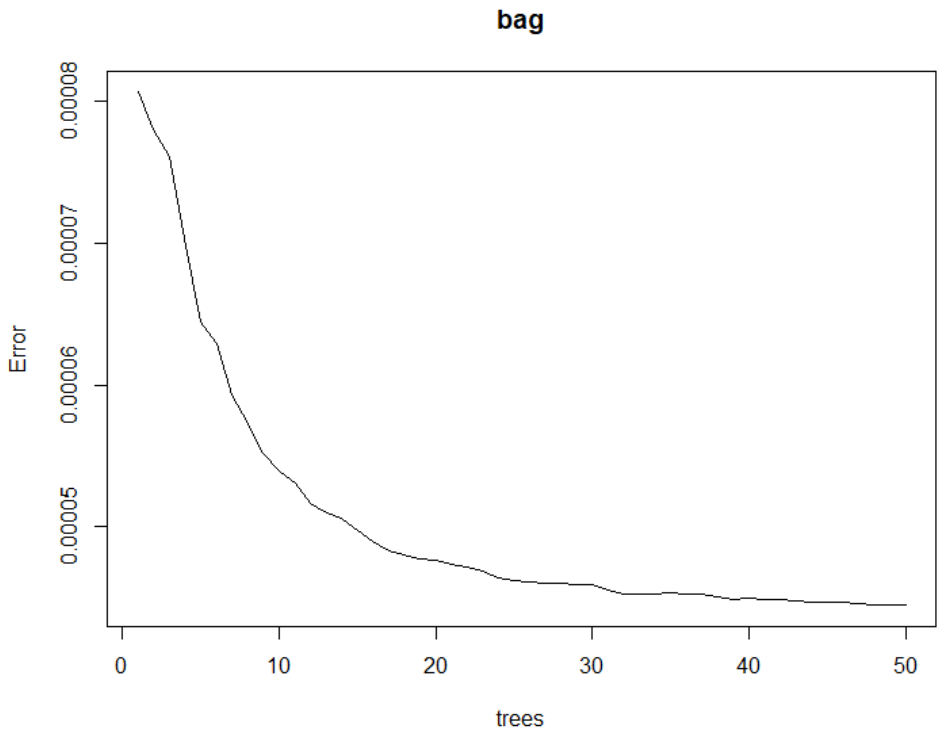

Рисунок 3.38 – Зменшення похибки під час виконання випадкового лісу

На рисунках 3.39, 3.40 наведені послідовність команд необхідних для прогнозування з застосуванням створеного випадкового лісу та тестової вибірки і побудови графіка залежності clicks від matched_requests.

```
> x <- test$matched_requests
> y <- predict(bag, newdata = test)
> x_y = data.frame(x, y)
> ggplot(train) +
+   geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
+   geom_smooth(data = data.frame(train$matched_requests, train$clicks), mapping = aes(x = train$matched_requests,
 y = train$clicks, color = "train smoothing")) +
+   geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
+   geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.39 – Застосування predict() і побудова першого графіка

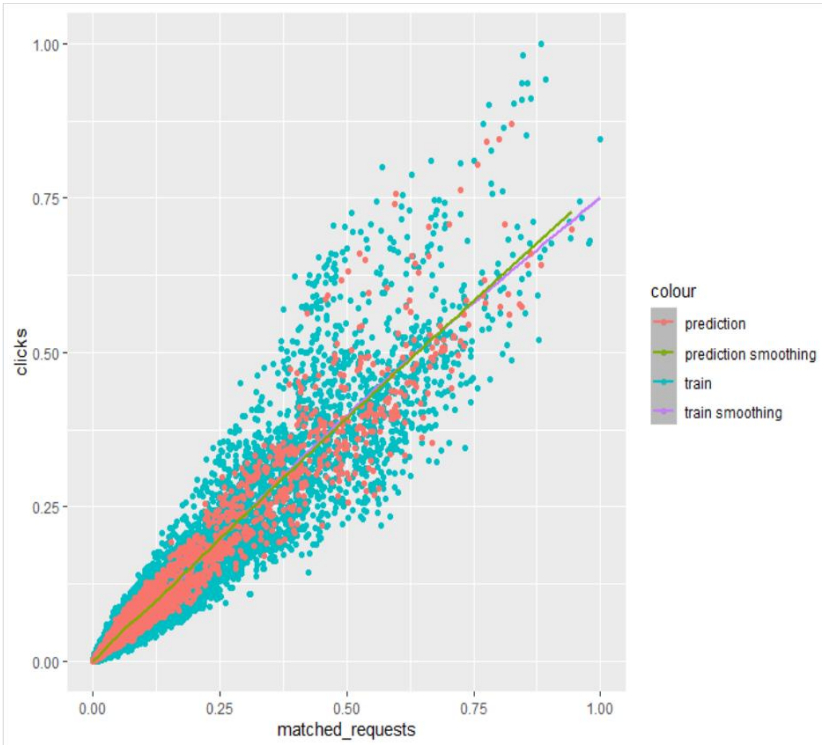

Рисунок 3.40 – Графік залежності значення clicks від matched_requests

Синіми маркерами позначені дані тренувальної вибірки, де вісь ордината представляє значення залежної змінної clicks, а абсциса – matched_requests. Червоні маркери позначають прогнозовані значення тестової вибірки. Лінії демонструють тренд зміни значень для даних тренувальної вибірки (фіолетовий колір) і тестової вибірки (зелений колір).

На рисунках 3.41, 3.42 наведено команди побудови другого графіка залежності clicks від impressions.

```
> y <- predict(bag, newdata = test)
> x <- test$impressions
> x_y = data.frame(x, y)
> ggplot(train) +
+   geom_point(aes(x = impressions, y = clicks, color = "train")) +
+   geom_smooth(data = data.frame(train$impressions, train$clicks), mapping = aes(x = train$impressions, y = trai
n$clicks, color = "train smoothing")) +
+   geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
+   geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.41 – Команди побудови другого графіка

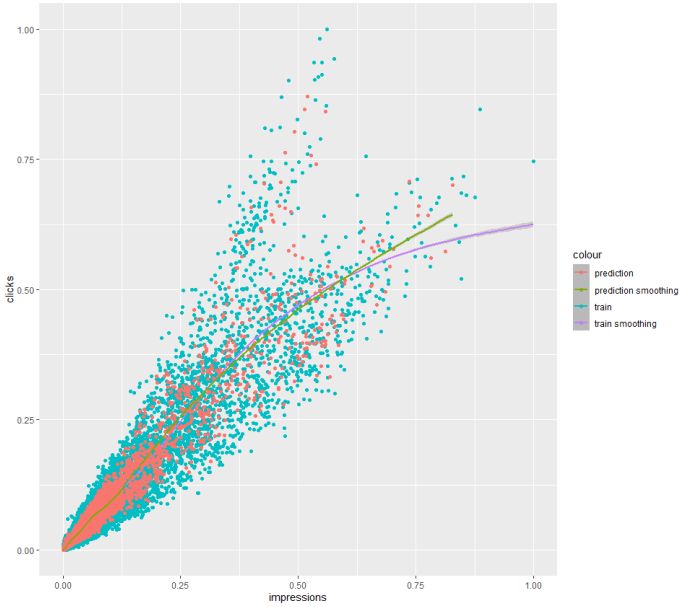

Рисунок 3.42 – Графік залежності значення clicks від impressions

Другий графік має позначення аналогічні до першого окрім того, що вісь абсцис позначає impressions.

Для оцінки точності прогнозування методом випадкового лісу, як і для регресійного дерева, застосовані метрики – MSE, MAE, RMSE. Для зручності порівняння, таблиця 3.4 містить метрики точності прогнозування методом випадкового лісу і регресійного дерева.

Таблиця 3.4 – Оцінка точності прогнозування

Модель	**MSE**	**MAE**	**RMSE**
Регресійне дерево	2208.425	14.64581	46.99388
Випадковий ліс	1663.3496	6.757985	40.78418

За результатами оцінки точності, а також графіків можна зробити висновок, що метод випадкового лісу дає значно кращі показники якості моделі ніж просте регресійне дерево.

3.6 Бустінг

Третім застосованим методом для прогнозування виступає бустінг, який також базується на використані декількох регресійних дерев, але на відміну від випадкового лісу дерева будуються послідовно, базуючись на попередньо створених деревах.

Бустінг є популярним алгоритмом машинного навчання, який успішно застосовується у багатьох сферах. У той час як випадковий ліс будує ансамбль глибоких дерев, бустінг створює ансамбль неглибоких і послідовних дерев, кожне з яких навчається і вдосконалюється на основі попереднього.

Використання даного методу базується на пакеті gbm, що є реалізацією розширень алгоритму AdaBoost Фройнда та Шапіра (Freund and Schapire) та gradient boosting machine Фрідмана (Friedman).

Алгоритм завантаження і підготовки даних відповідає застосованим раніше (рисунок 3.43).

R 4.2.1 · C:/RProject/

```
> library(randomForest)
> library(rsample)
> set.seed(2022)
> options(scipen=999)
> #завантаження набору даних
> data = read.csv("admob-report-modified.csv")
> #видалення зайвих змінних
> data$X = NULL
> data$country = NULL
> data$match_rate = NULL
> data$show_rate = NULL
> data$ctr = NULL
> data$app = NULL
> data$ad_source = NULL
> data$date = NULL
> #видалення викидів даних
> data = data[data$matched_requests < 100000,]
> data = data[data$impressions < 100000,]
> #нормалізація змінних
> process <- preProcess(as.data.frame(data), method=c("range"))
> data <- predict(process, as.data.frame(data))
```

Рисунок 3.43 – Завантаження набору даних для бустінгу

Як і раніше розбиття даних на тренувальну і тестову вибірки відбувається з застосуванням бібліотеки rsample, у пропорції 80% для тренувальної вибірки і відповідно 20% для тестової (рисунок 3.44).

```
> #розподіл набору даних на тестову і тренувальну вибірку
> split <- initial_split(data, 0.8)
> train <- training(split)
> test <- testing(split)
```

Рисунок 3.44 – Створення тренувальної і тестової вибірок

Для застосування бустінгу використовується функція gbm(). В якості параметрів передається залежна змінна, предиктори, тренувальний набір даних. Опція distribution приймає значення gaussian, оскільки вирішується задача прогнозування. Кількість дерев дорівнює 1000, максимальна глибина кожного дерева обмежена до 10. Також увімкнене відображення прогресу навчання, яке одразу починає працювати після виклику функції (рисунок 3.45).

```
> boost = gbm(clicks ~ requests + matched_requests + impressions, data = train, distribution = "gaussian", n.tree
s = 1000, interaction.depth = 10, verbose = TRUE, shrinkage = 0.001 )
Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        0.0014             nan     0.0010    0.0000
     2        0.0014             nan     0.0010    0.0000
     3        0.0014             nan     0.0010    0.0000
     4        0.0014             nan     0.0010    0.0000
     5        0.0014             nan     0.0010    0.0000
     6        0.0014             nan     0.0010    0.0000
```

Рисунок 3.45 – Застосування функції бустінгу

Виклик функції summary() після завершення навчання дозволяє переглянути відсотковий вплив змінних у текстовому (рисунок 3.46) і графічному (рисунок 3.47) вигляді.

```
> summary(boost)
                                 var   rel.inf
matched_requests matched_requests 92.822588
impressions           impressions  5.357534
requests                 requests  1.819877
```

Рисунок 3.46 – Відображення впливу змінних у текстовому вигляді

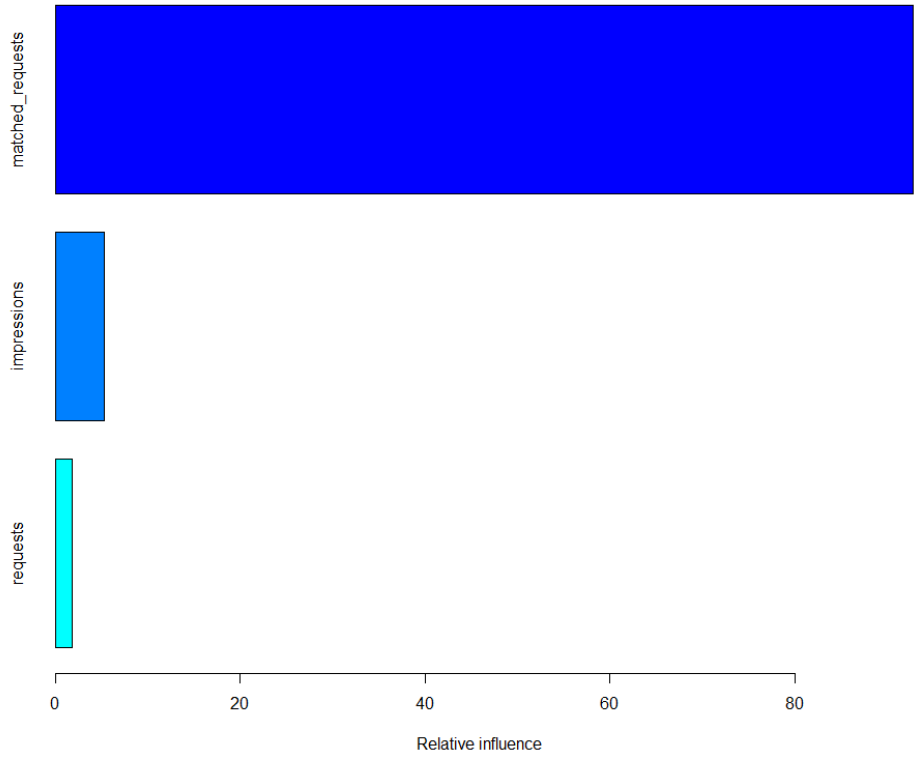

Рисунок 3.47 – Відображення впливу змінних у графічному вигляді

Наступний крок застосування створеної бустінг моделі для прогнозування на тестовій вибірці (рисунки 3.48, 3.49).

```
> x <- test$matched_requests
> y <- predict(boost, newdata = test, n.trees = 1000)
> x_y = data.frame(x, y)
> ggplot(train) +
+   geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
+   geom_smooth(data = data.frame(train$matched_requests, train$clicks), mapping = aes(x = train$matched_request
s, y = train$clicks, color = "train smoothing")) +
+   geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
+   geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.48 – Застосування predict()

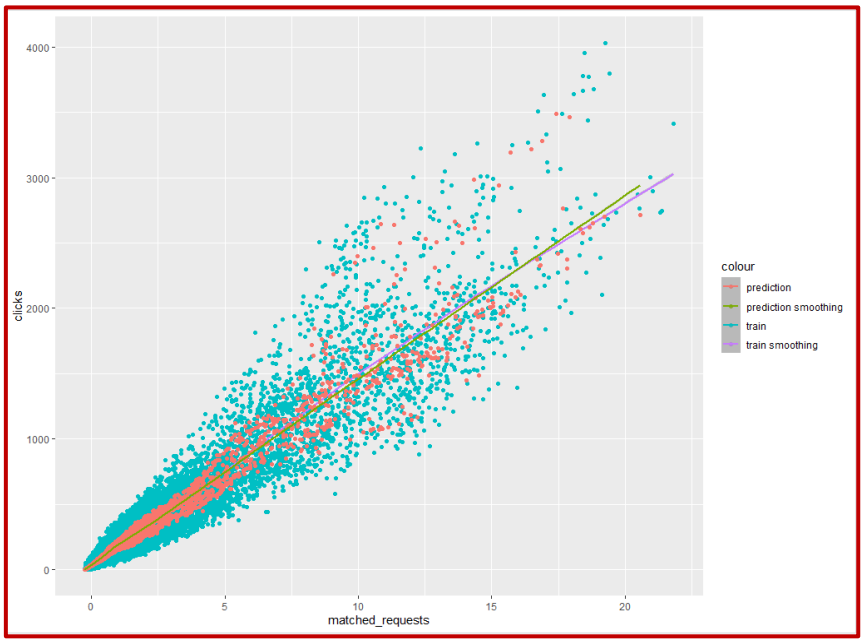

Рисунок 3.49 – Графік залежності значення clicks від matched_requests

Синіми маркерами позначені дані тренувальної вибірки, де вісь ордината представляє значення залежної змінної clicks, а абсциса – matched_requests. Червоні маркери позначають прогнозовані значення тестової вибірки. Лінії демонструють тренд зміни значень для даних тренувальної вибірки (фіолетовий колір) і тестової вибірки (зелений колір).

Для оцінки точності прогнозування методом бустінгу, як і для попередніх, застосовані метрики – MSE, MAE, RMSE. Для зручності порівняння, таблиця 3.5 містить метрики точності прогнозування бустінгу і попередніх методів.

Таблиця 3.5 – Оцінка точності прогнозування

Модель	**MSE**	**MAE**	**RMSE**
Регресійне дерево	2208.425	14.64581	46.99388
Випадковий ліс	1663.3496	6.757985	40.78418
Бустінг	1821.8466	7.816711	42.68309

Показники бустінгу значно кращі ніж у регресійного дерева, але трохи поступаються результатам випадкового лісу.

3.7 Нейронні мережі

Останнім розглянутим методом для вирішення задачі прогнозування ефективності реклами є створення моделей нейронної мережі, за допомогою інструментів TensorFlow for R і Keras API.

Алгоритм завантаження набору даних відповідає застосованим раніше (рисунок 3.50).

```
#завантаження набору даних
data = read.csv("admob-report-modified.csv")
#видалення зайвих змінних
data$X = NULL
data$country = NULL
data$match_rate = NULL
data$show_rate = NULL
data$ctr = NULL
data$app = NULL
data$ad_source = NULL
data$date = NULL
#видалення викидів даних
data = data[data$matched_requests < 100000,]
data = data[data$impressions < 100000,]
```

Рисунок 3.50 – Завантаження набору даних для нейронних мереж

Розбиття даних на тренувальну і тестову вибірки відбувається з застосуванням бібліотеки rsample, у пропорції 75% для тренувальної вибірки і 15% для тестової (рисунок 3.51).

```
#розподіл набору даних на тестову і тренувальну вибірку
split <- initial_split(data, 0.75)
train <- training(split)
test <- testing(split)
```

Рисунок 3.51 – Створення тренувальної і тестової вибірок

Додатковим кроком є відділення залежної змінної від предикторів. На рисунку 3.52 демонструється послідовність команди, що призводить до утворення двох дата фреймів з залежною змінною для тренувальної і тестової вибірок, а також двох дата фреймів з предикторами також для тренувальної і тестової вибірок.

```
#відділення залежної змінної від предикторів для тестових и навчальних даних
train_features <- train %>% select(-clicks)
test_features <- test %>% select(-clicks)
train_labels <- train %>% select(clicks)
test_labels <- test %>% select(clicks)
```

Рисунок 3.52 – Відокремлення залежної змінної від предикторів

Нормалізація відбувається шляхом створення спеціального шару у Keras API, який виступає вхідним для кожної моделі нейронної мережі (рисунок 3.53).

```
#створення функції для нормалізації даних у діапазоні від -1 до 1
normalizer <- layer_normalization(axis = -1L)
normalizer %>% adapt(as.matrix(train_features))
```

Рисунок 3.53 – Шар нормалізації

Для досягнення оптимального результату необхідно створити моделі нейронних мереж з різноманітними конфігураціями шарів, кількості нейронів та функцій активації.

Використовуючи функцію keras_model_sequential(), будуємо лінійну регресію з декількома входами, що складається з вхідного шару нормалізації і вихідного. У параметрах компіляції вказуються метрики оцінки точності моделі (рисунок 3.54).

```
#linear regression with multiple inputs
#налаштування параметрів моделі
model_1 <- keras_model_sequential() %>%
  normalizer() %>%
  layer_dense(units = 1)
model_1 %>% compile(
  optimizer = optimizer_adam(learning_rate = 0.1),
  loss = 'mean_squared_error',
  metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge')
)
```

Рисунок 3.54 – Налаштування і створення першої моделі нейронної мережі

Виклик функції Keras fit() розпочинає процес навчання протягом ста епох (рисунок 3.55). В об’єкті history зберігається статистика прогресу навчання, яку можливо візуалізувати за допомогою функції plot() (рисунок 3.56).

```
#навчання моделі
history <- model_1 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
  epochs = 100,
  # Calculate validation results on 50% of the training data.
  validation_split = 0.5
)
#метрики навчання
plot(history)
```

Рисунок 3.55 – Навчання першої моделі

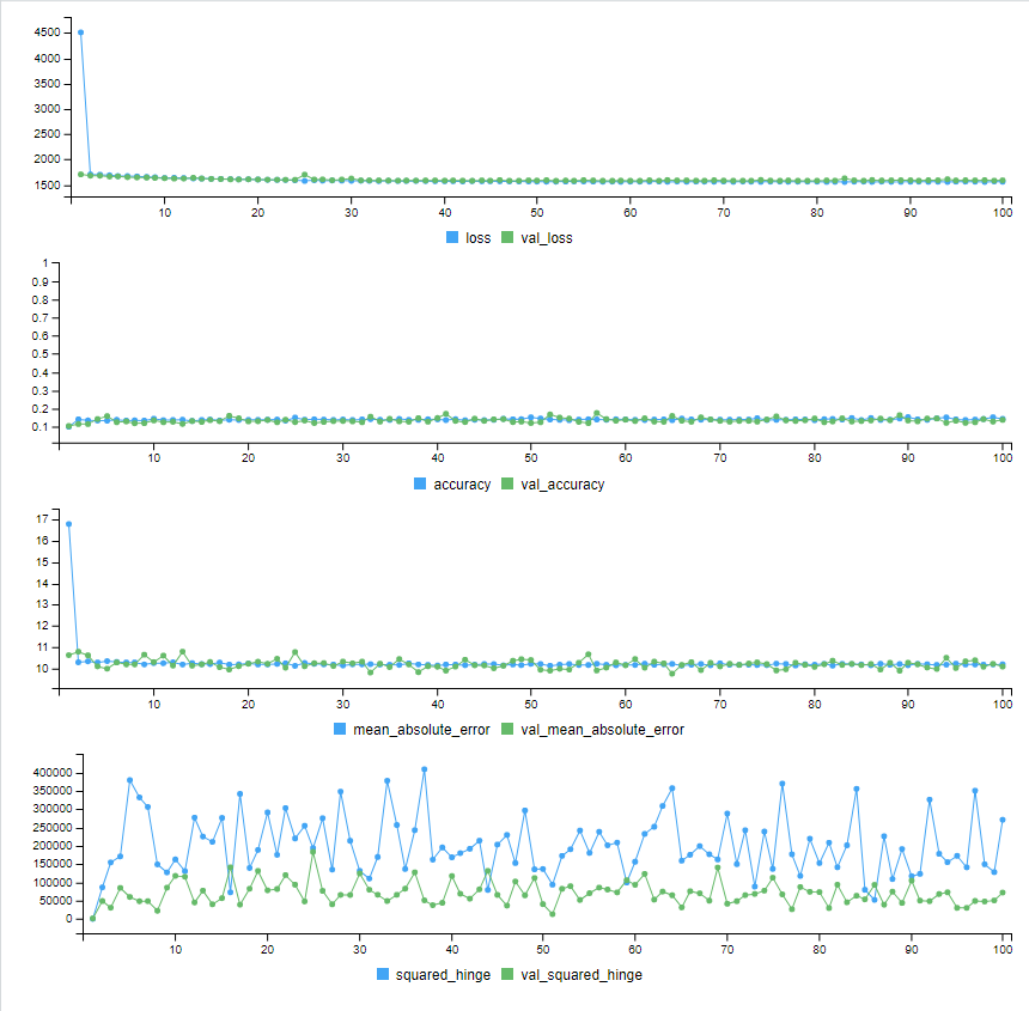

Рисунок 3.56 – Візуалізація статистики прогресу навчання першої моделі

На рисунку 3.57 наведено зведену інформацію про структуру першої моделі після закінчення навчання.

```
> #відображення структури
> model_1
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 normalization_1 (Normalization)  (None, 3)           7         
 dense_13 (Dense)            (None, 1)                 4         
=================================================================
Total params: 11
Trainable params: 4
Non-trainable params: 7
```

Рисунок 3.57 – Структура першої моделі

На рисунках 3.58, 3.59 наведені послідовність команд необхідних для прогнозування з застосуванням першої моделі та тестової вибірки і побудови графіка залежності clicks від matched_requests для першої моделі.

```
#тестування моделі, відображення залежності clicks від matched_requests
x <- test_features$matched_requests
y <- predict(model_1, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
              mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.58 – Тестування першої моделі і побудова графіка

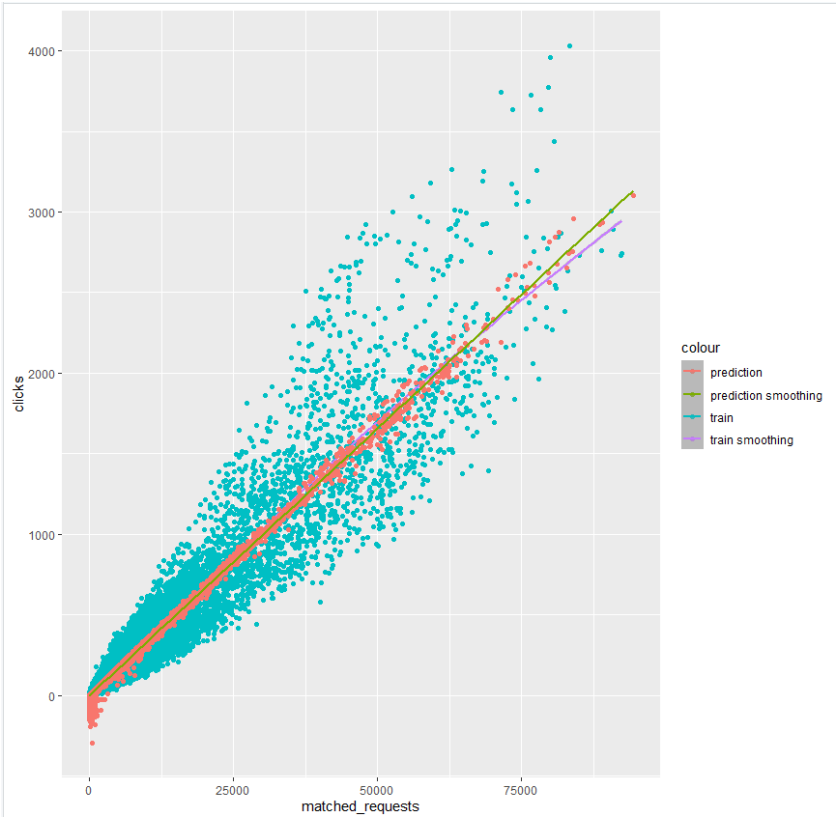

Рисунок 3.59 – Графік залежності значення clicks від matched_requests з застосуванням першої моделі

Синіми маркерами позначені дані тренувальної вибірки, де вісь ордината представляє значення залежної змінної clicks, а абсциса – matched_requests. Червоні маркери позначають прогнозовані значення тестової вибірки. Лінії демонструють тренд зміни значень для даних тренувальної вибірки (фіолетовий колір) і тестової вибірки (зелений колір).

На рисунках 3.60, 3.61 наведено команди побудови графіка залежності clicks від impressions.

```
#тестування моделі, відображення залежності clicks від impressions
x <- test_features$impressions
y <- predict(model_1, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
              mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.60 – Команди побудови графіка clicks від impressions

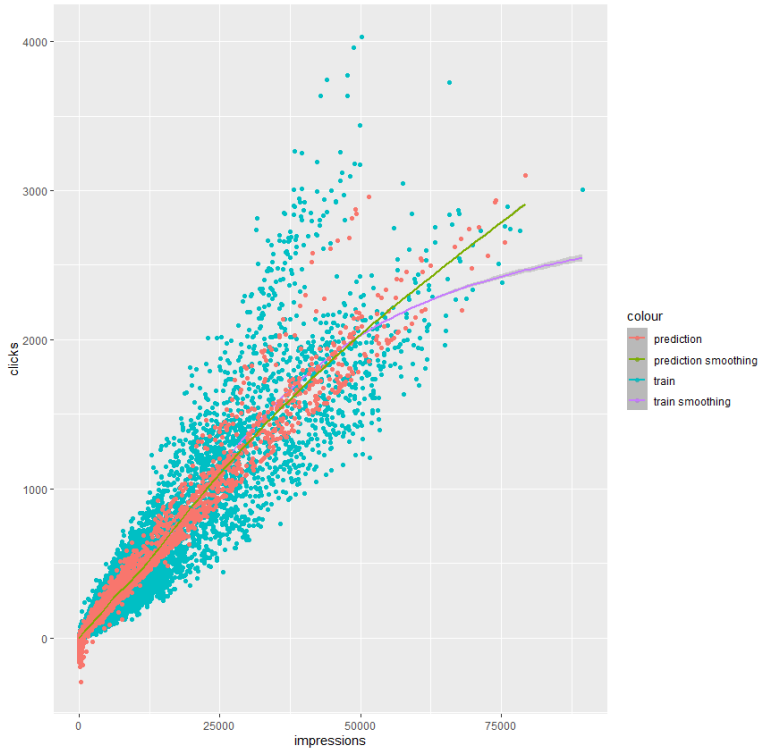

Рисунок 3.61 – Графік залежності значення clicks від impressions з застосуванням першої моделі

Графік має аналогічні до попереднього позначення окрім того, що вісь абсцис визначає impressions.

Для збереження метрик тестування моделі створюється окремий список, до якого буде записано метрики поточної і всіх наступних моделей. Значення метрик для першої моделі наведено на рисунку 3.62.

```
> #збереження метрик тестування моделі
> test_results <- list()
> test_results[['model_1']] <- model_1 %>%
+   evaluate(
+     as.matrix(test_features),
+     as.matrix(test_labels),
+     verbose = 0
+   )
> sapply(test_results, function(x) x)
                            model_1
loss                  1521.0217285
accuracy                 0.1371309
mean_absolute_error      9.9799366
squared_hinge        13083.7021484
```

Рисунок 3.62 – Оцінка якості прогнозування першої моделі

Для створення другої моделі визначається окрема функція яка, використовуючи keras_model_sequential(), будує та компілює регресійну модель глибокої нейронної мережі, що складається з вхідного шару нормалізації, двох прихованих шарів (64 нейрони, функція активації ReLU) і вихідного. У параметрах компіляції вказуються метрики оцінки точності моделі (рисунок 3.63).

```
#multiple-input DNN model 1
#налаштування параметрів моделі
build_and_compile_model <- function(norm) {
  model <- keras_model_sequential() %>%
    norm() %>%
    layer_dense(64, activation = 'relu') %>%
    layer_dense(64, activation = 'relu') %>%
    layer_dense(1)
  model %>% compile(
    loss = 'mean_squared_error',
    metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge'),
    optimizer = optimizer_adam(0.001)
  )
  model
}
model_2 <- build_and_compile_model(normalizer)
```

Рисунок 3.63 – Налаштування і створення другої моделі нейронної мережі

Виклик функції Keras fit() розпочинає процес навчання протягом ста епох (рисунок 3.64). В об’єкті history зберігається статистика прогресу навчання другої моделі, яку можливо візуалізувати за допомогою функції plot() (рисунок 3.65).

```
#навчання моделі
history <- model_2 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
  validation_split = 0.5,
  epochs = 100
)
#метрики навчання
plot(history)
```

Рисунок 3.64 – Навчання другої моделі

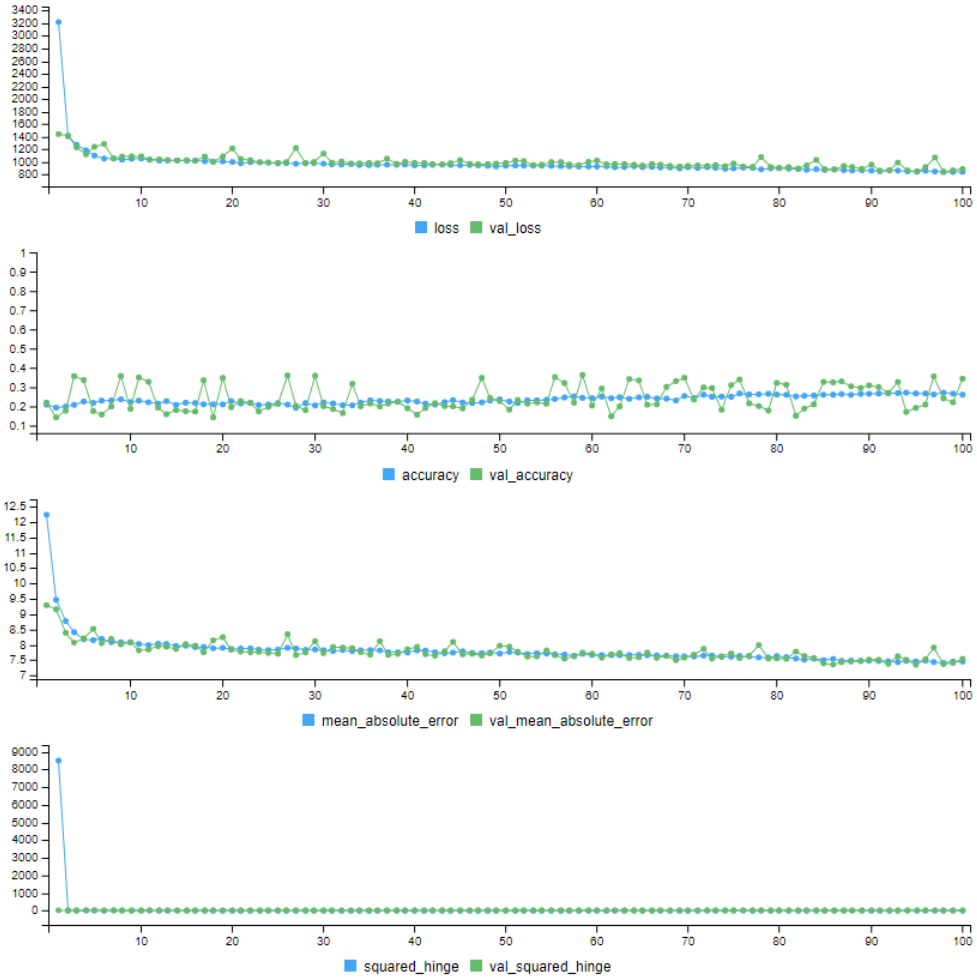

Рисунок 3.65 – Візуалізація статистики прогресу навчання другої моделі

На рисунку 3.66 наведено зведену інформацію про структуру другої моделі після закінчення навчання.

```
> #структура моделі
> model_2
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Trainable
=================================================================
 normalization_1 (Normalization) (None, 3)             7         Y
 dense_16 (Dense)            (None, 64)                256       Y
 dense_15 (Dense)            (None, 64)                4160      Y
 dense_14 (Dense)            (None, 1)                 65        Y
=================================================================
Total params: 4,488
Trainable params: 4,481
Non-trainable params: 7
```

Рисунок 3.66 – Структура другої моделі

На рисунках 3.67, 3.68 наведені команди для прогнозування з застосуванням другої моделі та тестової вибірки і побудови графіка залежності clicks від matched_requests для другої моделі.

```
#тестування моделі, відображення залежності clicks від matched_requests
x <- test_features$matched_requests
y <- predict(model_2, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
              mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.67 – Тестування другої моделі і побудова графіка

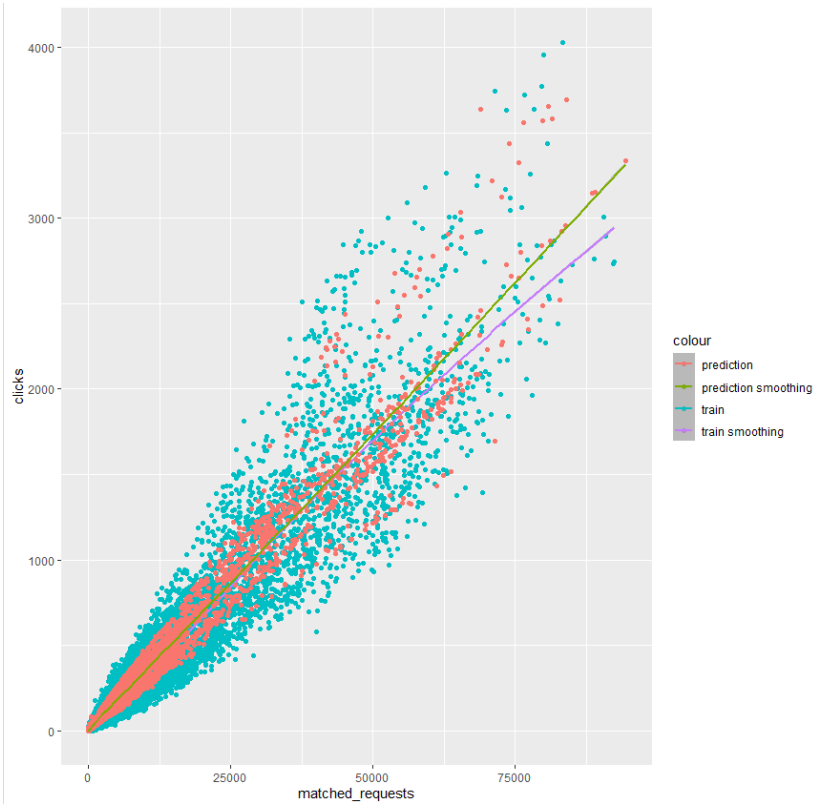

Рисунок 3.68 – Графік залежності значення clicks від matched_requests з застосуванням другої моделі

Синіми маркерами позначені дані тренувальної вибірки, де вісь ордината представляє значення залежної змінної clicks, а абсциса – matched_requests. Червоні маркери позначають прогнозовані значення тестової вибірки. Лінії демонструють тренд зміни значень для даних тренувальної вибірки (фіолетовий колір) і тестової вибірки (зелений колір).

На рисунках 3.69, 3.70 наведено команди побудови графіка залежності clicks від impressions для другої моделі.

```
#тестування моделі, відображення залежності clicks від impressions
x <- test_features$impressions
y <- predict(model_2, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
              mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.69 – Команди побудови графіка clicks від impressions з застосуванням другої моделі

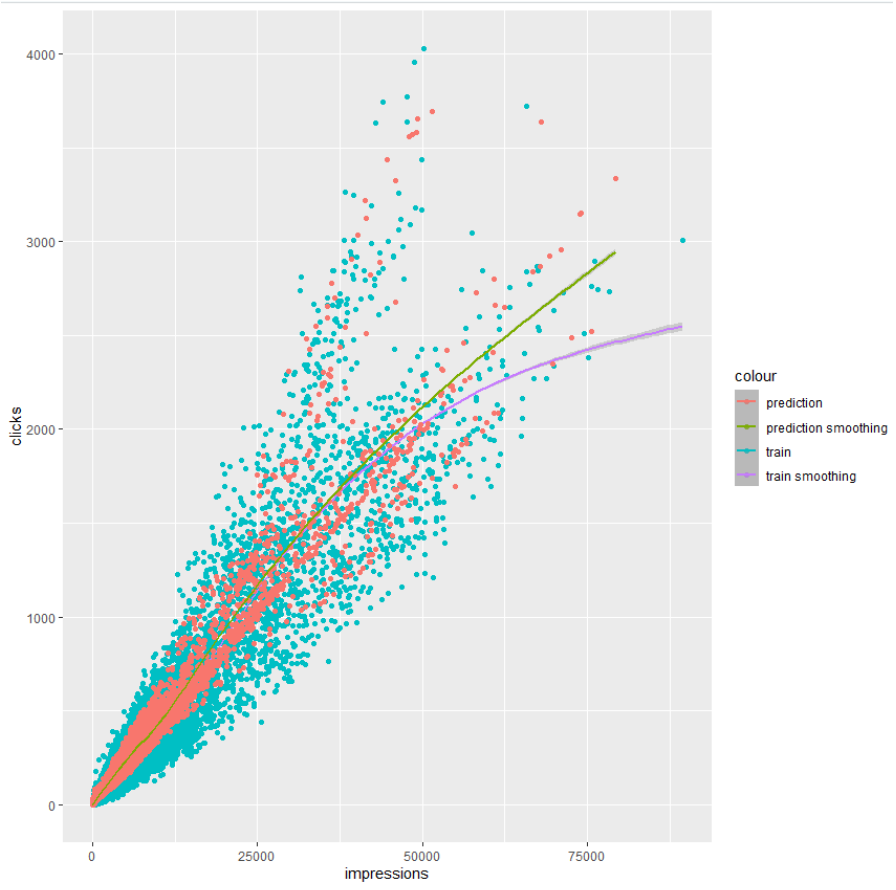

Рисунок 3.70 – Графік залежності значення clicks від impressions з застосуванням другої моделі

Графік має аналогічні до попереднього позначення окрім того, що вісь абсцис визначає impressions.

Значення метрик тестування другої моделі зберігається у раніше створений список (рисунок 3.71).

```
> #збереження метрик тестування
> test_results[['model_2']] <- model_2 %>% evaluate(
+   as.matrix(test_features),
+   as.matrix(test_labels),
+   verbose = 0
+ )
> sapply(test_results, function(x) x)
                         model_1     model_2
loss                1521.0217285 835.9135132
accuracy               0.1371309   0.3454359
mean_absolute_error    9.9799366   7.4311185
squared_hinge      13083.7021484   0.3899768
```

Рисунок 3.71 – Оцінка якості прогнозування другої моделі

Для створення третьої моделі визначається нова функція яка, використовуючи keras_model_sequential(), будує та компілює регресійну модель глибокої нейронної мережі, що складається з вхідного шару нормалізації, двох прихованих шарів (64 нейрони, функція активації Sigmoid; 128 нейронів, функція активації Sigmoid) і вихідного. У параметрах компіляції вказуються метрики оцінки точності моделі (рисунок 3.72).

```
#multiple-input DNN model 2
#налаштування параметрів моделі
build_and_compile_model <- function(norm) {
  model <- keras_model_sequential() %>%
    norm() %>%
    layer_dense(64, activation = 'sigmoid') %>%
    layer_dense(128, activation = 'sigmoid') %>%
    layer_dense(1)
  model %>% compile(
    loss = 'mean_squared_error',
    metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge'),
    optimizer = optimizer_adam(0.001)
  )
  model
}
model_3 <- build_and_compile_model(normalizer)
```

Рисунок 3.72 – Налаштування і створення третьої моделі нейронної мережі

Виклик функції Keras fit() розпочинає процес навчання протягом ста епох (рисунок 3.73). В об’єкті history зберігається статистика прогресу навчання

третьої моделі, яку можливо візуалізувати за допомогою функції plot() (рисунок 3.74).

```
#навчання моделі
history <- model_3 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
  validation_split = 0.5,
  epochs = 100
)
#метрики навчання
plot(history)
```

Рисунок 3.73 – Навчання третьої моделі

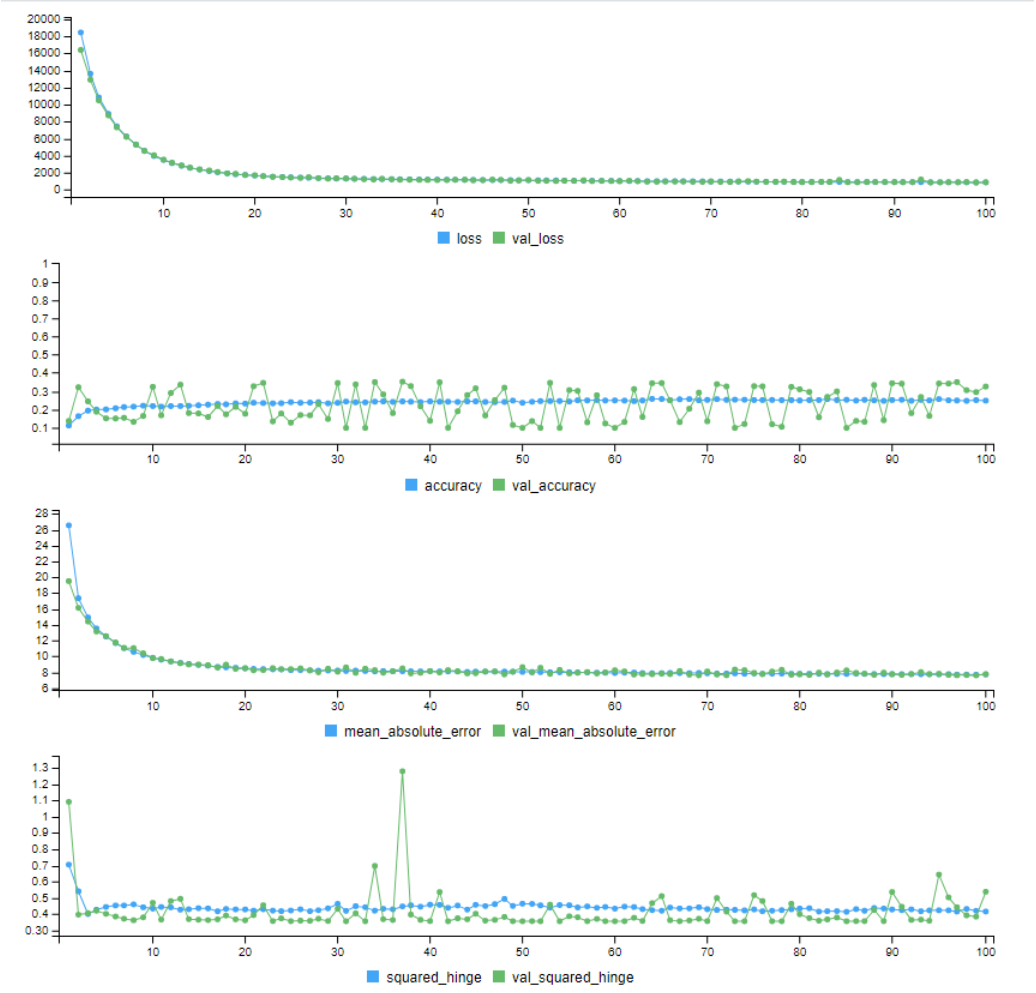

Рисунок 3.74 – Візуалізація статистики прогресу навчання третьої моделі

На рисунку 3.75 наведено зведену інформацію про структуру третьої моделі після закінчення навчання.

```
> #структура моделі
> model_3
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 normalization_1 (Normalization)  (None, 3)           7         
 dense_19 (Dense)            (None, 64)                256       
 dense_18 (Dense)            (None, 128)               8320      
 dense_17 (Dense)            (None, 1)                 129       
=================================================================
Total params: 8,712
Trainable params: 8,705
Non-trainable params: 7
```

Рисунок 3.75 – Структура третьої моделі

На рисунках 3.76, 3.78 наведені команди для прогнозування з застосуванням третьої моделі та тестової вибірки і побудови графіка залежності clicks від matched_requests для третьої моделі.

```
#тестування моделі, відображення залежності clicks від matched_requests
x <- test_features$matched_requests
y <- predict(model_3, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
              mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.76 – Тестування третьої моделі і побудова графіка

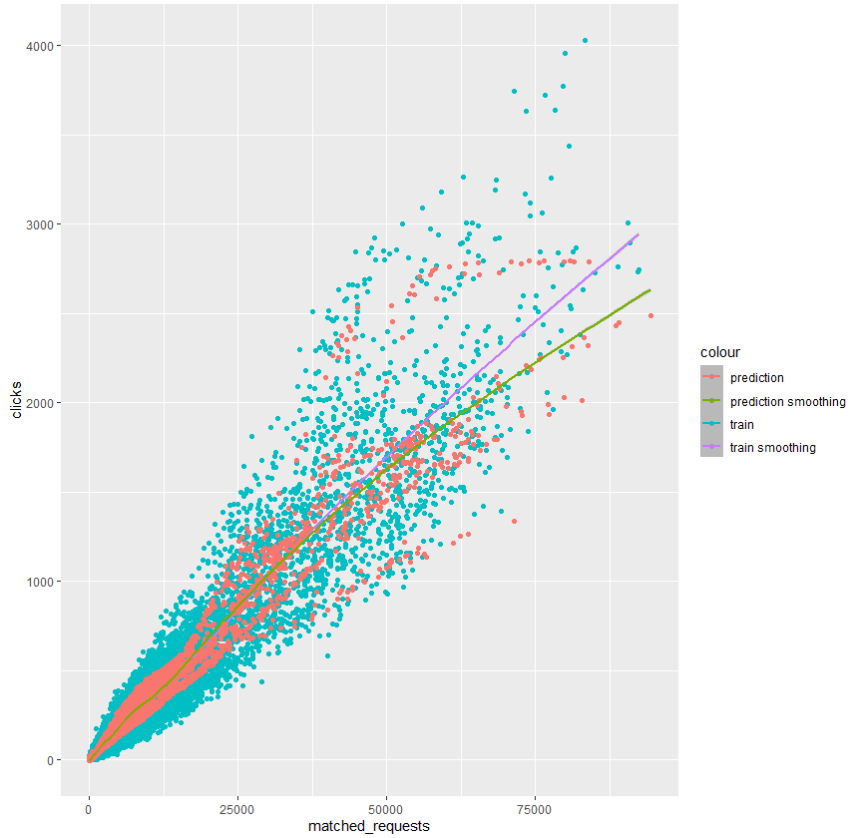

Рисунок 3.78 – Графік залежності значення clicks від matched_requests з застосуванням третьої моделі

Синіми маркерами позначені дані тренувальної вибірки, де вісь ордината представляє значення залежної змінної clicks, а абсциса – matched_requests. Червоні маркери позначають прогнозовані значення тестової вибірки. Лінії демонструють тренд зміни значень для даних тренувальної вибірки (фіолетовий колір) і тестової вибірки (зелений колір).

На рисунках 3.79, 3.80 наведено команди побудови графіка залежності clicks від impressions для третьої моделі.

```
#тестування моделі, відображення залежності clicks від impressions
x <- test_features$impressions
y <- predict(model_3, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
              mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.79 – Команди побудови графіка clicks від impressions з застосуванням третьої моделі

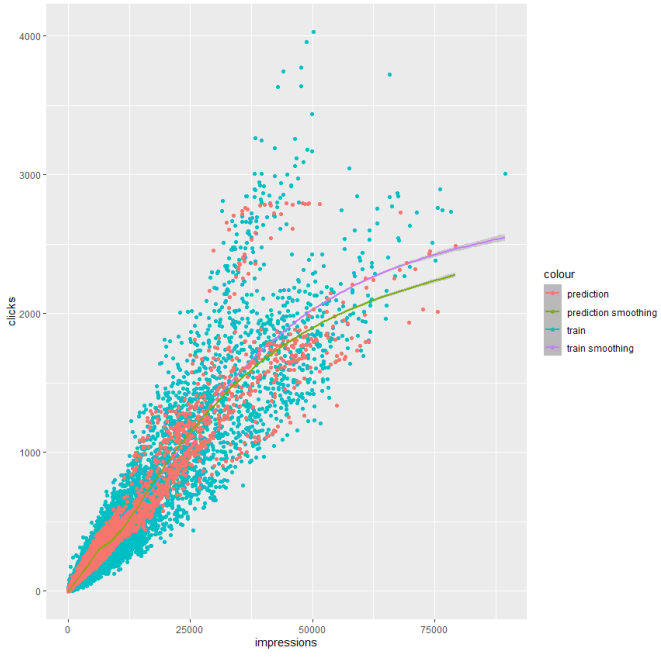

Рисунок 3.80 – Графік залежності значення clicks від impressions з застосуванням третьої моделі

Графік має аналогічні до попереднього позначення окрім того, що вісь абсцис визначає impressions.

Значення метрик тестування другої моделі зберігається у раніше створений список (рисунок 3.81).

```
> #збереження метрик тестування моделі
> test_results[['model_3']] <- model_3 %>% evaluate(
+   as.matrix(test_features),
+   as.matrix(test_labels),
+   verbose = 0
+ )
> sapply(test_results, function(x) x)
                         model_1     model_2     model_3
loss                1521.0217285 835.9135132 922.4628906
accuracy               0.1371309   0.3454359   0.3242998
mean_absolute_error    9.9799366   7.4311185   7.7538595
squared_hinge      13083.7021484   0.3899768   0.5172967
```

Рисунок 3.81 – Оцінка якості прогнозування третьої моделі

Для створення четвертої моделі також застосовується нова функція яка, використовуючи Keras API, будує та компілює регресійну модель глибокої нейронної мережі, що складається з вхідного шару нормалізації, одного прихованого шару (128 нейрони, функція активації ReLu) і вихідного. У параметрах компіляції вказуються метрики оцінки точності моделі (рисунок 3.82).

```
#multiple-input DNN model 3
#налаштування параметрів моделі
build_and_compile_model <- function(norm) {
  model <- keras_model_sequential() %>%
    norm() %>%
    layer_dense(128, activation = 'relu') %>%
    layer_dense(1)
  model %>% compile(
    loss = 'mean_squared_error',
    metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge'),
    optimizer = optimizer_adam(0.001)
  )
  model
}
model_4 <- build_and_compile_model(normalizer)
```

Рисунок 3.82 – Налаштування і створення четвертої моделі нейронної мережі

Виклик функції Keras fit() розпочинає процес навчання протягом ста епох (рисунок 3.83). В об’єкті history зберігається статистика прогресу навчання четвертої моделі, яку можливо візуалізувати за допомогою функції plot() (рисунок 3.84).

```
#навчання моделі
history <- model_4 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
  validation_split = 0.5,
  epochs = 100
)
#метрики навчання
plot(history)
```

Рисунок 3.83 – Навчання четвертої моделі

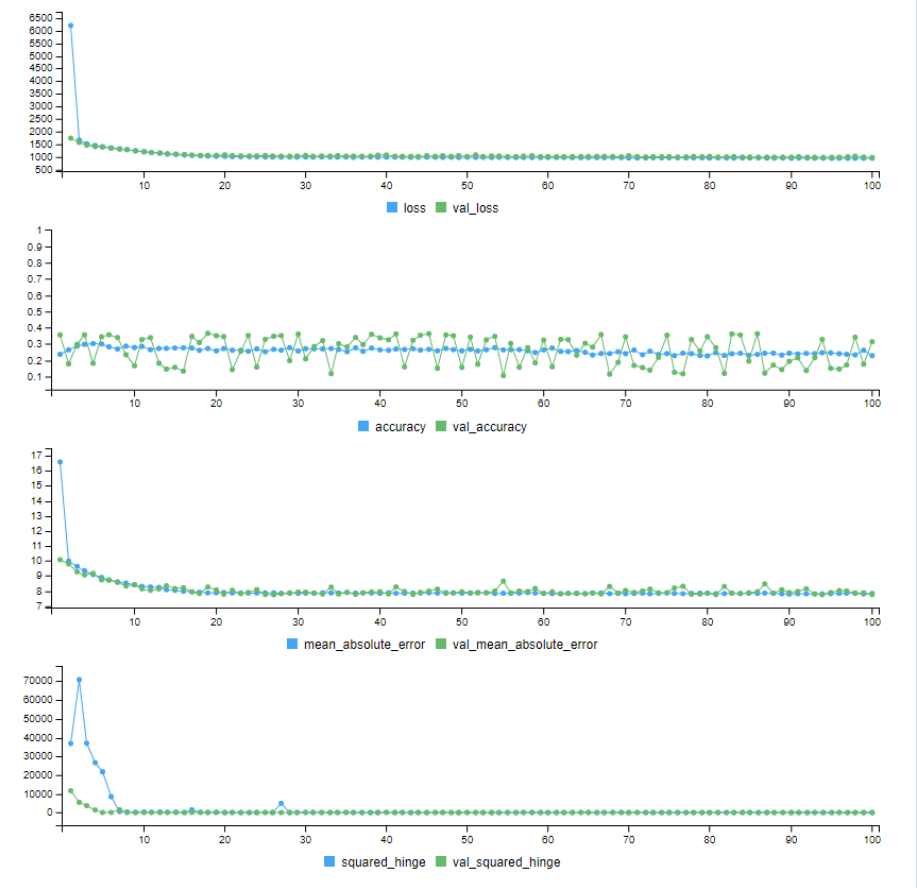

Рисунок 3.84 – Візуалізація статистики прогресу навчання третьої моделі

На рисунку 3.85 наведено зведену інформацію про структуру четвертої моделі після закінчення навчання.

```
> #структура моделі
> model_4
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Trainable
=================================================================
 normalization_1 (Normalization) (None, 3)             7         Y
 dense_21 (Dense)            (None, 128)               512       Y
 dense_20 (Dense)            (None, 1)                 129       Y
=================================================================
Total params: 648
Trainable params: 641
Non-trainable params: 7
_________________________________________________________________
```

Рисунок 3.85 – Структура четвертої моделі

На рисунках 3.86, 3.87 наведені команди для прогнозування з застосуванням четвертої моделі та тестової вибірки і побудови графіка залежності clicks від matched_requests для четвертої моделі.

```
#тестування моделі, відображення залежності clicks від matched_requests
x <- test_features$matched_requests
y <- predict(model_4, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
              mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.86 – Тестування четвертої моделі і побудова графіка

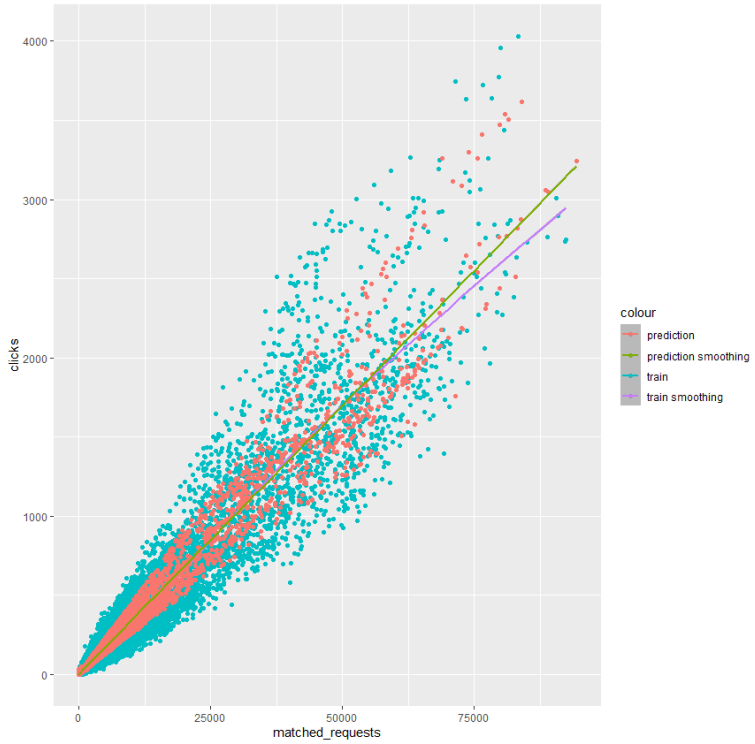

Рисунок 3.87 – Графік залежності значення clicks від matched_requests з застосуванням четвертої моделі

Синіми маркерами позначені дані тренувальної вибірки, де вісь ордината представляє значення залежної змінної clicks, а абсциса – matched_requests. Червоні маркери позначають прогнозовані значення тестової вибірки. Лінії демонструють тренд зміни значень для даних тренувальної вибірки (фіолетовий колір) і тестової вибірки (зелений колір).

На рисунках 3.88, 3.89 наведено команди побудови графіка залежності clicks від impressions для четвертої моделі.

```
#тестування моделі, відображення залежності clicks від impressions
x <- test_features$impressions
y <- predict(model_4, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
              mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
```

Рисунок 3.88 – Команди побудови графіка clicks від impressions з застосуванням четвертої моделі

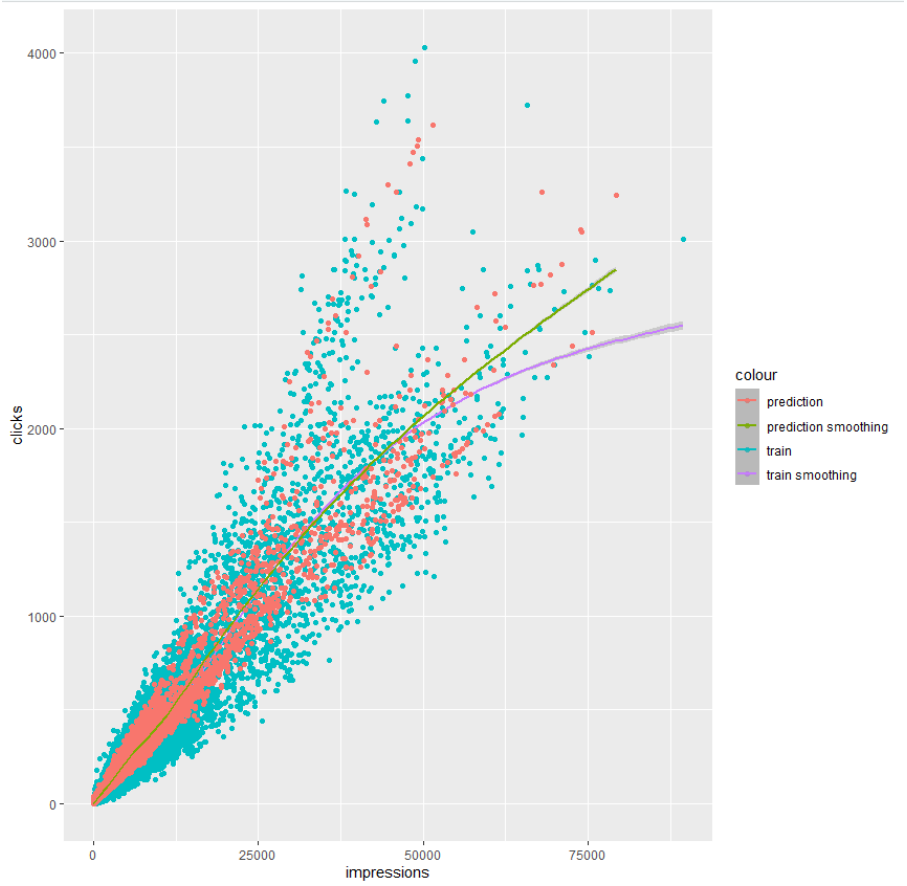

Рисунок 3.89 – Графік залежності значення clicks від impressions з застосуванням четвертої моделі

Графік має аналогічні до попереднього позначення окрім того, що вісь абсцис визначає impressions.

Значення метрик тестування четвертої моделі зберігається у раніше створений список (рисунок 3.90).

```
> #збереження метрик тестування моделі
> test_results[['model_4']] <- model_4 %>% evaluate(
+   as.matrix(test_features),
+   as.matrix(test_labels),
+   verbose = 0
+ )
> sapply(test_results, function(x) x)
                         model_1     model_2     model_3     model_4
loss                1521.0217285 835.9135132 922.4628906 960.5924683
accuracy               0.1371309   0.3454359   0.3242998   0.3127043
mean_absolute_error    9.9799366   7.4311185   7.7538595   7.6298661
squared_hinge      13083.7021484   0.3899768   0.5172967   0.5198945
> |
```

Рисунок 3.90 – Оцінка якості прогнозування четвертої моделі

Для наочного порівняння точності прогнозування створених моделей необхідно скласти зведену таблицю.

Таблиця 3.6 – Оцінювання моделей нейронних мереж

DNN моделі			**Метрики оцінки якості прогнозування**		
Назва	**Структура**	**Функція активації**	**MSE**	**MAE**	**RMSE**
model_1	3; 1		1521.021728	9.979937	39.000278
model_5	3; 32; 1	ReLU	997.197632	7.877985	31.578436
model_6	3; 64; 1	ReLU	1151.843872	8.308035	33.938825
model_7	3; 32; 32; 1	Sigmoid	17776.564453	38.533802	133.328783
model_8	3; 32; 64; 1	Sigmoid	21166.892578	51.711899	145.488462
model_9	3; 64; 32; 1	ReLU	948.718811	7.771630	30.801279
model_2	3; 64; 64; 1	ReLU	835.913513	7.431118	28.912169
model_3	3; 64; 128; 1	Sigmoid	922.462891	7.753859	30.372074
model_10	3; 128; 64; 1	ReLU	1036.494995	8.169644	32.194642
model_11	3; 128; 128; 1	ReLU	1063.928467	8.110294	32.617916
model_4	3; 128; 1	ReLU	960.592468	7.629866	30.993426
model_12	3; 32; 64; 32; 1	ReLU	1122.228027	8.1411247	33.499672

В результаті серед усіх створених моделей, найкращою за всіма показниками є нейронна мережа model_2, що складається з двох прихованих шарів, кожен з яких має 64 нейрони і функцію активації ReLU.

Висновки до розділу 3

Реалізація підходу до прогнозування, що описана в даному розділі є фінальним етапом всієї роботи, а також тісно пов’язана з результатами роботи минулих розділів.

В ході роботи було проведено підготовку і дослідження набору даних, це дозволило визначити ціль прогнозування, а також алгоритми для досягнення цього.

Для прогнозування застосовано чотири методи машинного навчання, а саме: регресійне дерево, випадковий ліс, бустінг та нейронні мережі. У таблиці 3.7 наведено результати оцінювання якості прогнозування для кожного з використаних методів.

Таблиця 3.7 – Порівняльна таблиця якості прогнозу моделей

Модель	**MSE**	**MAE**	**RMSE**
Регресійне дерево	2208.425	14.64581	46.99388
Випадковий ліс	1663.3496	6.757985	40.78418
Бустінг	1821.8466	7.816711	42.68309
model_2	835.9135132	7.4311185	28.9121689

Остаточно можна зробити висновок, що найкращим методом прогнозування є створення нейронної мережі, з урахуванням налаштувань параметрів задля підвищення точності. Друге місце посідає випадковий ліс, трохи гірше за нього бустінг. Найгірші показники має регресійне дерево.

ВИСНОВКИ

Під час написання магістерської кваліфікаційної роботи проведено аналіз сфери цифрового маркетингу, його інтеграцію у сучасні пристрої та застосунки, а також розвиток технологій штучного інтелекту і машинного навчання для застосування в якості інструменті маркетингу.

Головною метою роботи було підвищення ефективності аналізу і прогнозування показників рекламних інтернет-застосунків методами машинного навчання. Оскільки застосування методів машинного навчання у маркетингу дає змогу швидко отримати статистику та відкоригувати параметри націлювання та розміщення реклами для отримання найбільшої ефективності.

У результаті проведеної роботи були вирішені наступні завдання:

1) проаналізовано існуючи сервіси для прогнозування показників реклами;

2) головним інструментальним засобом для аналізу обрано середовище і мову програмування R;

3) вибір регресійного дерева, випадкового лісу, бустінгу та нейронних мереж як найпопулярніших методів для вирішення проблеми прогнозування показників реклами;

4) реалізовано підхід до прогнозування, застосовуючи методи машинного навчання;

5) проведено оцінку їх ефективності і покращено якість прогнозування шляхом коригування параметрів.

У методичній частині роботи опрацьовано матеріал практичного завдання, що може стати частиною навчальної дисципліни «Методи та системи машинного навчання». У спеціальній частині з охорони праці вирішені проблеми при роботі з серверним обладнанням в офісному приміщенні і складено план дій для персоналу під час надзвичайних ситуацій.

Тому завершення перерахованих завдань свідчить про успішне досягнення поставленої мети даної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. History of mobile phones and the first mobile phone : вебсайт. URL: https://www.uswitch.com/mobiles/guides/history-of-mobile-phones (дата звернення: 20.01.2023).

2. Jan Rutkowski. The evolution of mobile devices : вебсайт. URL: https://billennium.com/blog/the-evolution-of-mobile-devices (дата звернення: 20.01.2023).

3. Natalie Lynn. The History and Evolution of Mobile Advertising : вебсайт. URL: https://gimbal.com/history-evolution-mobile-advertising (дата звернення: 20.01.2023).

4. The Evolution of Marketing Using Mobile Apps : вебсайт. URL: https://worldfinancialreview.com/the-evolution-of-marketing-using-mobile-apps (дата звернення: 20.01.2023).

5. Alex Cocotas. The Mobile Advertising Ecosystem Explained : вебсайт. URL: https://www.businessinsider.com/the-mobile-advertising-system-explained-2012-10 (дата звернення: 20.01.2023).

6. Mapping the Mobile Advertising Ecosystem : вебсайт. URL: https://www.appsflyer.com/resources/ecommerce/mapping-the-mobile-advertising-ecosystem-cheat-sheet (дата звернення: 20.01.2023).

7. Chaitanya Chandrasekar. What Is Predictive Advertising & Why Do You Need It? : вебсайт. URL: https://www.searchenginejournal.com/predictive-advertising-need/248427 (дата звернення: 20.01.2023).

8. IBM Watson Advertising Create opportunity with data and AI : вебсайт. URL: https://www.ibm.com/watson-advertising (дата звернення: 20.01.2022).

9. Google Ads : вебсайт. URL: https://developers.google.com/google-ads (дата звернення: 20.01.2023).

10. Microsoft Ad Supply Forecasts : вебсайт. URL: https://learn.microsoft.com/en-us/linkedin/marketing/integrations/ads/advertising-targeting/ad-supply-forecasts (дата звернення: 20.01.2023).

11. What is R? : вебсайт. URL: https://www.r-project.org/about.html (дата звернення: 20.01.2023).

12. A (Brief) History of R : вебсайт. URL: https://mran.microsoft.com/documents/what-is-r (дата звернення: 20.01.2023).

13. R packages on CRAN and Bioconductor : вебсайт. URL: https://www.rdocumentation.org (дата звернення: 20.01.2023).

14. An Introduction to corrplot Package : вебсайт. URL: https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html (дата звернення: 20.01.2023).

15. Support Functions and Datasets for Venables and Ripley's MASS : вебсайт. URL: https://www.rdocumentation.org/packages/MASS/versions/7.3-58.2 (дата звернення: 20.01.2023).

16. Fit a Classification or Regression Tree : вебсайт. URL: https://www.rdocumentation.org/packages/tree/versions/1.0-43/topics/tree (дата звернення: 20.01.2023).

17. General Resampling Infrastructure : вебсайт. URL: https://rsample.tidymodels.org (дата звернення: 20.01.2023).

18. Ben Hamner [aut, cph], Michael Frasco [aut, cre], Erin LeDell [ctb]. Evaluation Metrics for Machine Learning : документація. CRAN, 2018. 26 с. URL: https://cran.r-project.org/web/packages/Metrics/Metrics.pdf (дата звернення: 20.01.2023).

19. Machine Learning Evaluation Metrics : вебсайт. URL: https://www.rdocumentation.org/packages/MLmetrics/versions/1.1.1 (дата звернення: 20.01.2023).

20. Classification And REgression Training : вебсайт. URL: https://topepo.github.io/caret (дата звернення: 20.01.2023).

21. Classification and Regression with Random Forest : вебсайт. URL: https://www.rdocumentation.org/packages/randomForest/versions/4.7-1.1/topics/randomForest (дата звернення: 20.01.2023).

22. Generalized Boosted Regression Modeling (GBM) : вебсайт. URL: https://www.rdocumentation.org/packages/gbm/versions/2.1.8.1/topics/gbm (дата звернення: 20.01.2023).

23. R Interface to Python : вебсайт. URL: https://rstudio.github.io/reticulate (дата звернення: 20.01.2023).

24. R packages for data science : вебсайт. URL: https://www.tidyverse.org (дата звернення: 20.01.2023).

25. TensorFlow history : вебсайт. URL: https://en.wikipedia.org/wiki/TensorFlow (дата звернення: 20.01.2023).

26. Introduction to Tensors : вебсайт. URL: https://www.tensorflow.org/guide/tensor (дата звернення: 20.01.2023).

27. Keras the Python deep learning API : вебсайт. URL: https://keras.io (дата звернення: 20.01.2023).

28. Mean Squared Error : вебсайт. URL: https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_528 (дата звернення: 20.01.2023).

29. Mean Absolute Error : вебсайт. URL: https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_525 (дата звернення: 20.01.2023).

30. Root-mean-square deviation : вебсайт. URL: https://en.wikipedia.org/wiki/Root-mean-square_deviation (дата звернення: 20.01.2023).

31. Fish market. Database of common fish species for fish market : вебсайт. URL: https://www.kaggle.com/datasets/aungpyaeap/fish-market (дата звернення: 20.01.2023).

32. TensorFlow for R. Quick start : вебсайт. URL: https://tensorflow.rstudio.com/install (дата звернення: 20.01.2023).

33. Releford, Dallas 2000. Setting up a server room, part 1: The basics. TechRepublic.: веб-сайт. URL: http://www.techrepublic.com/article/setting-up-a-serverroom-part-1-the-basics/1052501 (дата звернення: 20.01.2023).

34. Higbie, Carrie 2005. Rules for designing the urban data center.: веб-сайт. URL: http://searchdatacenter.techtarget.com/tip/Rules-for-designing-the-urbandata-center?topic=300039 (дата звернення: 20.01.2023).

35. DiMinico, Chris 2006. Telecommunications Infrastructure. Standard for Data Centers. ANSI/TIA-942.: веб-сайт. URL: http://www.ieee802.org/3/hssg/public/nov06/diminico_01_1106.pdf (дата звернення: 20.01.2023).

36. Bicsi. ANSI/BICSI 002-2011, Data Center Design and Implementation Best Practices.: веб-сайт. URL: https://www.bicsi.org/book_details.aspx?Book=BICSI-002-CM-11-v2&d=0 (дата звернення: 20.01.2023).

37. Nygaard, Stein 2010. Requirements for the design of ICT rooms. Best Practice Document. Produced by UNINETT led working group on physical infrastructure (No UFS103).: веб-сайт. URL: http://www.terena.org/activities/campus-bp/pdf/gn3-na3-t4-ufs103.pdf (дата звернення: 20.01.2023).

38. DiMinico, Chris 2006. Telecommunications Infrastructure. Standard for Data Centers. ANSI/TIA-942.: веб-сайт. URL: http://www.ieee802.org/3/hssg/public/nov06/diminico_01_1106.pdf (дата звернення: 20.01.2023).

39. McFarlane, Robert 2007. What is the minimum ceiling height in a data center?: веб-сайт. URL: https://searchdatacenter.techtarget.com/feature/What-is-the-minimum-ceiling-height-in-a-data-center#:~:text=Expert%20Robert%20Macfarlane%20offers%20some%20advice.&te

xt=The%20short%20answer%3A%20at%20least,its%20mixing%20with%20cold%20air (дата звернення: 20.01.2023).

40. PTS Data Center Solutions. Data Center Design. Computer Room Flooring.: веб-сайт. URL: http://www.ptsdcs.com/comproomflooring.asp (дата звернення: 20.01.2023).

41. OpenExtra 2005. Calculating the Size of a Server Room Air Conditioner.: веб-сайт. URL: http://www.openxtra.co.uk/articles/calculating-heat-load (дата звернення: 20.01.2023).

42. Steveinator. How to build or redesign a server room. SpiceWorks.: веб-сайт. URL: http://community.spiceworks.com/how_to/show/2622-how-to-build-orredesign-a-server-room (дата звернення: 20.01.2023).

43. Sustainable Computing. Best Practices: Server Rooms.: веб-сайт. URL: https://sustainablecomputing.umich.edu/knowledge/best-practices.php (дата звернення: 20.01.2023).

44. Elkins Technologies 2011. LED Lighting Cuts Cooling Costs in IT Room.: веб-сайт. URL: http://elkinstech.com/?p=8 (дата звернення: 20.01.2023).

45. Siemens. Aspirating smoke detection for areas requiring very high sensitivity.: веб-сайт. URL: https://www.downloads.siemens.com/download-center/Download.aspx?pos=download&fct=getasset&id1=A6V10583600 (дата звернення: 20.01.2023).

46. Concept Fire Suppression Ltd. Inert Gas Fire Suppression.: веб-сайт. URL: http://www.conceptfire-uk.com/products/inert-gas-ig55-inergen-ig541/ (дата звернення: 20.01.2023).

ДОДАТОК А

Код реалізації прогнозування методами машинного навчання

```
#завантаження набору даних з файлу
data = read.csv("admob-report.csv")
#видалення порядкового номеру
data$X = NULL
#усі значення записів у стовпцях повязаних з аукціоном = 0
summary(data$Bid.requests)
#видалення стовпця
data$Bid.requests = NULL
summary(data$Bids.in.auction)
#видалення стовпця
data$Bids.in.auction = NULL
summary(data$Winning.bids)
#видалення стовпця
data$Winning.bids = NULL
#зміна назв стовпців
names(data)
names(data)[1] = 'app'
names(data)[2] = 'ad_source'
names(data)[3] = 'country'
names(data)[4] = 'date'
names(data)[5] = 'requests'
names(data)[6] = 'match_rate'
names(data)[7] = 'matched_requests'
names(data)[8] = 'show_rate'
names(data)[9] = 'impressions'
names(data)[10] = 'ctr'
names(data)[11] = 'clicks'
names(data)
#перетворення тексту у формат дати
data$date <- as.Date(data$date)
#перетворення текстових відсотків у числовий формат з застосуванням нормалізації
data$match_rate = as.numeric(sub("%" , "", data$match_rate)) / 100
data$show_rate = as.numeric(sub("%" , "", data$show_rate)) / 100
data$ctr = as.numeric(sub("%" , "", data$ctr)) / 100
#перевірка коректності даних
summary(data$date)
summary(data$requests)
```

```
summary(data$match_rate)
summary(data$matched_requests)
# show_rate містить 25380 NA's
summary(data$show_rate)
#show_rate NA fix
#відокремлення NA записів від решти даних
dataIncorrectShowRate = data[is.na(data$show_rate),]
data = data[!is.na(data$show_rate),]
#послідовний обхід всіх записів, що містять NA
for(i in 1:nrow(dataIncorrectShowRate)) {
  #витягнення окремого рядку-запису за поточним індексом
  row = dataIncorrectShowRate[i,]
  #якщо show_rate = NA то необхідно встановити значення 0 або перерахувати
  if (!is.na(row$show_rate)) next
  #show_rate = impressions / matched_requests
  if (row$impressions == 0 || row$matched_requests == 0)
    row$show_rate = 0
  else
    row$show_rate = row$impressions / row$matched_requests
  #збереження модифікованого значення
  dataIncorrectShowRate[i,] = row
}
#включення модифікованих записів до основного дата фрейму
data = rbind(data, dataIncorrectShowRate)
#видалення тимчасової змінної
dataIncorrectShowRate = NULL
summary(data$impressions)
# ctr містить 42520 NA's
summary(data$ctr)
#ctr NA fix
#відокремлення NA записів від решти даних
dataIncorrectCtr = data[is.na(data$ctr),]
data = data[!is.na(data$ctr),]
#послідовний обхід всіх записів, що містять NA
for(i in 1:nrow(dataIncorrectCtr)) {
  #витягнення окремого рядку-запису за поточним індексом
  row = dataIncorrectCtr[i,]
  #якщо ctr = NA то необхідно встановити значення 0 або перерахувати
  if (!is.na(row$ctr)) next
  # ctr = clicks / impressions
```

```
  if (row$clicks == 0 || row$impressions == 0)
    row$ctr = 0
  else
    row$ctr = row$clicks / row$impressions
  #збереження модифікованого значення
  dataIncorrectCtr[i,] = row
}
#включення модифікованих записів до основного дата фрейму
data = rbind(data, dataIncorrectCtr)
#видалення тимчасової змінної
dataIncorrectCtr = NULL
# Max = 8, CTR не може бути більше 1 (100%) => видаляємо все що більше 1
summary(data$ctr)
data = data[data$ctr <= 1,]
summary(data$clicks)
#збереження дата фрейму у файл
write.csv(data,"admob-report-modified.csv", row.names = TRUE)

install.packages("corrplot")
library(corrplot)
install.packages("psych")
library(psych)
#завантаження набору даних з файлу
data = read.csv("admob-report-modified.csv")
#видалення порядкового номеру
data$X = NULL
#відображення структури набору даних
str(data)
#видалення стовпців
data$app = NULL
data$ad_source = NULL
data$country = NULL
data$date = NULL
summary(data$clicks)
hist(data$clicks)
boxplot(data$clicks)
cor = cor(data)
corrplot(cor)
#побудова розширеної матриці кореляції
pairs = pairs(~requests+match_rate+matched_requests+show_rate+impressions+ctr+clicks,data=data)
```

```
pairs.panels(data)


install.packages("MASS")
install.packages("tree")
install.packages('Metrics')
install.packages('MLmetrics')
library(MASS)
library(tree)
library(rsample)
library(Metrics)
library(MLmetrics)
library(caret)
set.seed(2022)
options(scipen=999)
#завантаження набору даних
data = read.csv("admob-report-modified.csv")
#видалення зайвих змінних
data$X = NULL
data$country = NULL
data$match_rate = NULL
data$show_rate = NULL
data$ctr = NULL
data$app = NULL
data$ad_source = NULL
data$date = NULL
#видалення викидів даних
data = data[data$matched_requests < 100000,]
data = data[data$impressions < 100000,]
#нормалізація змінних
process <- preProcess(as.data.frame(data), method=c("range"))
data <- predict(process, as.data.frame(data))
#розподіл набору даних на тестову і тренувальну вибірку
split <- initial_split(data, 0.8)
train <- training(split)
test <- testing(split)
#створення дерева регресії
tree = tree(clicks ~ requests + matched_requests + impressions, train)
summary(tree)
plot(tree)
text(tree, pretty = 0)
```

```
cv = cv.tree(tree)
plot(cv$size, cv$dev, type = 'b')
x <- test$matched_requests
y <- predict(tree, test)
x_y = data.frame(x, y)
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train$matched_requests, train$clicks), mapping = aes(x = train$matched_requests, y = train$clicks, color = "train smoothing")) +
  geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
x <- test$requests
x_y = data.frame(x, y)
ggplot(train) +
  geom_point(aes(x = requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train$requests, train$clicks), mapping = aes(x = train$requests, y = train$clicks, color = "train smoothing")) +
  geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
x <- test$impressions
x_y = data.frame(x, y)
ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train$impressions, train$clicks), mapping = aes(x = train$impressions, y = train$clicks, color = "train smoothing")) +
  geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
ggplot(data.frame(pred = as.numeric(y), clicks = test$clicks)) +
  geom_point(aes(x = pred, y = clicks)) +
  geom_abline(intercept = 0, slope = 1, color = "blue")
#calculate MSE
mean((test$clicks - y)^2)
#calculate MAE
mae(test$clicks, y)
#calculate RMSE
sqrt(mean((test$clicks - y)^2))

install.packages("randomForest")
library(randomForest)
library(rsample)
```

```
set.seed(2022)
options(scipen=999)
#завантаження набору даних
data = read.csv("admob-report-modified.csv")
#видалення зайвих змінних
data$X = NULL
data$country = NULL
data$match_rate = NULL
data$show_rate = NULL
data$ctr = NULL
data$app = NULL
data$ad_source = NULL
data$date = NULL
#видалення викидів даних
data = data[data$matched_requests < 100000,]
data = data[data$impressions < 100000,]
#нормалізація змінних
process <- preProcess(as.data.frame(data), method=c("range"))
data <- predict(process, as.data.frame(data))
#розподіл набору даних на тестову і тренувальну вибірку
split <- initial_split(data, 0.8)
train <- training(split)
test <- testing(split)
bag = randomForest(clicks ~ requests + matched_requests + impressions, data = train, mtry = 3, importance = TRUE, ntree = 50)
bag
importance(bag)
plot(bag)
x <- test$matched_requests
y <- predict(bag, newdata = test)
x_y = data.frame(x, y)
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train$matched_requests, train$clicks), mapping = aes(x = train$matched_requests, y = train$clicks, color = "train smoothing")) +
  geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
x <- test$impressions
x_y = data.frame(x, y)
ggplot(train) +
```

```
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train$impressions, train$clicks), mapping = aes(x = train$impressions, y = train$clicks, color = "train smoothing")) +
  geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
#calculate MSE
mean((test$clicks - y)^2)
#calculate MAE
mae(test$clicks, y)
#calculate RMSE
sqrt(mean((test$clicks - y)^2))



install.packages("gbm")
library(gbm)
library(rsample)
set.seed(2022)
options(scipen=999)
#завантаження набору даних
data = read.csv("admob-report-modified.csv")
#видалення зайвих змінних
data$X = NULL
data$country = NULL
data$match_rate = NULL
data$show_rate = NULL
data$ctr = NULL
data$app = NULL
data$ad_source = NULL
data$date = NULL
#видалення викидів даних
data = data[data$matched_requests < 100000,]
data = data[data$impressions < 100000,]
#нормалізація змінних
process <- preProcess(as.data.frame(data), method=c("range"))
data <- predict(process, as.data.frame(data))
#розподіл набору даних на тестову і тренувальну вибірку
split <- initial_split(data, 0.8)
train <- training(split)
test <- testing(split)
```

```
boost = gbm(clicks ~ requests + matched_requests + impressions, data = train, distribution = "gaussian", n.trees = 1000, interaction.depth = 10, verbose = TRUE, shrinkage = 0.001 )
summary(boost)
boost
importance(boost)
plot(boost)
par(mfrow = c(1, 2))
plot(boost, i = "matched_requests")
plot(boost, i = "impressions")
plot(boost, i = "requests")
x <- test$matched_requests
y <- predict(boost, newdata = test, n.trees = 1000)
x_y = data.frame(x, y)
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train$matched_requests, train$clicks), mapping = aes(x = train$matched_requests, y = train$clicks, color = "train smoothing")) +
  geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
x <- test$impressions
x_y = data.frame(x, y)
ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train$impressions, train$clicks), mapping = aes(x = train$impressions, y = train$clicks, color = "train smoothing")) +
  geom_point(data = x_y,  mapping = aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = x_y, mapping = aes(x = x, y = y, color = "prediction smoothing"))
#calculate MSE
mean((test$clicks - y)^2)
#calculate MAE
mae(test$clicks, y)
#calculate RMSE
sqrt(mean((test$clicks - y)^2))



install.packages("MASS")
install.packages("recipes")
install.packages("rsample")
install.packages("tidyverse")
install.packages("tensorflow")
```

```
library(reticulate)
path_to_python <- "C:/ProgramData/Miniconda3/python.exe"
 #install_python()
virtualenv_create("r-reticulate", python = path_to_python)
virtualenv_list()
virtualenv_remove("r-reticulate2")
install_tensorflow(method = "conda", conda = "C:/ProgramData/Miniconda3/_conda.exe")
install_tensorflow(method = "conda")
library(tensorflow)
install_tensorflow(envname = "r-reticulate")
tf$constant("Hello Tensorflow!")
tf$config$list_physical_devices("GPU")
library(MASS)
library(recipes)
library(rsample)
library(keras)
library(tidyverse)
options(scipen=999)
#завантаження набору даних
data = read.csv("admob-report-modified.csv")
#видалення зайвих змінних
data$X = NULL
data$country = NULL
data$match_rate = NULL
data$show_rate = NULL
data$ctr = NULL
data$app = NULL
data$ad_source = NULL
data$date = NULL
#видалення викидів даних
data = data[data$matched_requests < 100000,]
data = data[data$impressions < 100000,]
#розподіл набору даних на тестову і тренувальну вибірку
split <- initial_split(data, 0.75)
train <- training(split)
test <- testing(split)
#відділення залежної змінної від предикторів для тестових и навчальних даних
train_features <- train %>% select(-clicks)
test_features <- test %>% select(-clicks)
train_labels <- train %>% select(clicks)
```

```
test_labels <- test %>% select(clicks)
#створення функції для нормалізації даних у діапазоні від -1 до 1
normalizer <- layer_normalization(axis = -1L)
normalizer %>% adapt(as.matrix(train_features))
#linear regression with multiple inputs
#налаштування параметрів моделі
model_1 <- keras_model_sequential() %>%
  normalizer() %>%
  layer_dense(units = 1)
model_1 %>% compile(
  optimizer = optimizer_adam(learning_rate = 0.1),
  loss = 'mean_squared_error',
  metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge')
)
#навчання моделі
history <- model_1 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
  epochs = 100,
  # Calculate validation results on 50% of the training data.
  validation_split = 0.5
)
#метрики навчання
plot(history)
#відображення структури
model_1
#тестування моделі, відображення залежності clicks від matched_requests
x <- test_features$matched_requests
y <- predict(model_1, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
              mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train
smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
#тестування моделі, відображення залежності clicks від impressions
x <- test_features$impressions
y <- predict(model_1, as.matrix(test_features))
ggplot(train) +
```

```
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
              mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
#тестування моделі, відображення порівняння значень тестових значень clicks від прогнозованих
test_predictions_1 <- predict(model_1, as.matrix(test_features))
ggplot(data.frame(pred = as.numeric(test_predictions_1), clicks = test_labels$clicks)) +
  geom_point(aes(x = pred, y = clicks)) +
  geom_abline(intercept = 0, slope = 1, color = "blue")
#збереження метрик тестування моделі
test_results <- list()
test_results[['model_1']] <- model_1 %>%
  evaluate(
    as.matrix(test_features),
    as.matrix(test_labels),
    verbose = 0
  )
sapply(test_results, function(x) x)
##################
#multiple-input DNN model 1
#налаштування параметрів моделі
build_and_compile_model <- function(norm) {
  model <- keras_model_sequential() %>%
    norm() %>%
    layer_dense(64, activation = 'relu') %>%
    layer_dense(64, activation = 'relu') %>%
    layer_dense(1)
  model %>% compile(
    loss = 'mean_squared_error',
    metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge'),
    optimizer = optimizer_adam(0.001)
  )
  model
}
model_2 <- build_and_compile_model(normalizer)
#навчання моделі
history <- model_2 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
```

```
  validation_split = 0.5,
  epochs = 100
 )
 #метрики навчання
 plot(history)
 #структура моделі
 model_2
 #тестування моделі, відображення залежності clicks від matched_requests
 x <- test_features$matched_requests
 y <- predict(model_2, as.matrix(test_features))
 ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
        mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
 #тестування моделі, відображення залежності clicks від impressions
 x <- test_features$impressions
 y <- predict(model_2, as.matrix(test_features))
 ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
        mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
 #тестування моделі, відображення порівняння значень тестових значень clicks від прогнозованих
 test_predictions_2 <- predict(model_2, as.matrix(test_features))
 ggplot(data.frame(pred = as.numeric(test_predictions_2), clicks = test_labels$clicks)) +
  geom_point(aes(x = pred, y = clicks)) +
  geom_abline(intercept = 0, slope = 1, color = "blue")
 #збереження метрик тестування
 test_results[['model_2']] <- model_2 %>% evaluate(
  as.matrix(test_features),
  as.matrix(test_labels),
  verbose = 0
 )
 sapply(test_results, function(x) x)
 ##################
 #multiple-input DNN model 2
```

```
#налаштування параметрів моделі
build_and_compile_model <- function(norm) {
  model <- keras_model_sequential() %>%
    norm() %>%
    layer_dense(64, activation = 'sigmoid') %>%
    layer_dense(128, activation = 'sigmoid') %>%
    layer_dense(1)
  model %>% compile(
    loss = 'mean_squared_error',
    metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge'),
    optimizer = optimizer_adam(0.001)
  )
  model
}
model_3 <- build_and_compile_model(normalizer)
#навчання моделі
history <- model_3 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
  validation_split = 0.5,
  epochs = 100
)
#метрики навчання
plot(history)
#структура моделі
model_3
#тестування моделі, відображення залежності clicks від matched_requests
x <- test_features$matched_requests
y <- predict(model_3, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
              mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
#тестування моделі, відображення залежності clicks від impressions
x <- test_features$impressions
y <- predict(model_3, as.matrix(test_features))
ggplot(train) +
```

```
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
              mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
#тестування моделі, відображення порівняння значень тестових значень clicks від прогнозованих
test_predictions_3 <- predict(model_3, as.matrix(test_features))
ggplot(data.frame(pred = as.numeric(test_predictions_3), clicks = test_labels$clicks)) +
  geom_point(aes(x = pred, y = clicks)) +
  geom_abline(intercept = 0, slope = 1, color = "blue")
#збереження метрик тестування моделі
test_results[['model_3']] <- model_3 %>% evaluate(
  as.matrix(test_features),
  as.matrix(test_labels),
  verbose = 0
)
sapply(test_results, function(x) x)
##################
#multiple-input DNN model 3
#налаштування параметрів моделі
build_and_compile_model <- function(norm) {
  model <- keras_model_sequential() %>%
    norm() %>%
    layer_dense(128, activation = 'relu') %>%
    layer_dense(1)
  model %>% compile(
    loss = 'mean_squared_error',
    metrics=list('accuracy', 'mean_absolute_error', 'squared_hinge'),
    optimizer = optimizer_adam(0.001)
  )
  model
}
model_4 <- build_and_compile_model(normalizer)
#навчання моделі
history <- model_4 %>% fit(
  as.matrix(train_features),
  as.matrix(train_labels),
  validation_split = 0.5,
  epochs = 100
)
```

```
#метрики навчання
plot(history)
#структура моделі
model_4
#тестування моделі, відображення залежності clicks від matched_requests
x <- test_features$matched_requests
y <- predict(model_4, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = matched_requests, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$matched_requests, train_labels$clicks),
              mapping = aes(x = train_features$matched_requests, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
#тестування моделі, відображення залежності clicks від impressions
x <- test_features$impressions
y <- predict(model_4, as.matrix(test_features))
ggplot(train) +
  geom_point(aes(x = impressions, y = clicks, color = "train")) +
  geom_smooth(data = data.frame(train_features$impressions, train_labels$clicks),
              mapping = aes(x = train_features$impressions, y = train_labels$clicks, color = "train smoothing")) +
  geom_point(data = data.frame(x, y), aes(x = x, y = y, color = "prediction")) +
  geom_smooth(data = data.frame(x, y), mapping = aes(x = x, y = y, color = "prediction smoothing"))
#тестування моделі, відображення порівняння значень тестових значень clicks від прогнозованих
test_predictions_4 <- predict(model_4, as.matrix(test_features))
ggplot(data.frame(pred = as.numeric(test_predictions_4), clicks = test_labels$clicks)) +
  geom_point(aes(x = pred, y = clicks)) +
  geom_abline(intercept = 0, slope = 1, color = "blue")
#збереження метрик тестування моделі
test_results[['model_4']] <- model_4 %>% evaluate(
  as.matrix(test_features),
  as.matrix(test_labels),
  verbose = 0
)
sapply(test_results, function(x) x)
```