Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**Створення Recommendation System модель для оптимізації бюджету на Apple Search Ads шляхом управління bid-ставок**»

Виконав: студент 4-го року навчання
освітньої програми «Прикладна
математика»,
спеціальності 113 Прикладна
математика
Курдюков Дмитро Сергійович
Керівник: Дрінь С.С.,
кандидат фіз.-мат. наук, ст. викладач
Рецензент: Артеменко Л.П.
кандидат екон. наук, доцент

Кваліфікаційна робота захищена
з оцінкою _____
Секретар ЕК _____
«____» _____ 20____ р.

Київ – 2022

ЗМІСТ

Тема: Створення Recommendation System модель для оптимізації бюджету на Apple Search Ads шляхом управління bid-ставок.

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на класифікаційну роботу.	6.10.2021	
2.	Огляд технічної літератури за темою роботи.	13.11.2021	
3.	Опрацювання матеріалів	01.12.2022	
4.	Погодження плану робіт з керівником	22.12.2022	
5.	Розв’язання поставленої задачі	25.04.2022	
6.	Розробка комп’ютерної реалізації алгоритмів	20.05.2022	
7.	Виконання порівняльного аналізу результатів прогнозування	25.05.2022	
8.	Створення слайдів та написання доповіді	15.06.2022	
9.	Остаточне оформлення роботи	19.06.2022	
10.	Захист кваліфікаційної роботи	4.07.2022	

Студент: Курдюков Д.С. _____

Керівник: Дрінь С.С._____

“6” жовтня 2021 р

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
проф., д.ф.-м.н.
_____М. М. Глибовець

(підпис)
„____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на класифікаційну роботу

студенту Курдюкову Дмитру Сергійовичу факультету інформатики 4 курсу

ТЕМА Створення Recommendation System модель для оптимізації бюджету на Apple Search Ads шляхом управління bid-ставок.

Зміст ТЧ до класифікаційної роботи:

Індивідуальне завдання

Вступ

1 Теоретичні відомості з дерев, випадкового лісу, перехресної перевірки та Apple Search Ads

2 Розробка моделі

Висновки

Список літератури

Додатки

Дата видачі „6” жовтня 2021 р. Керівник _____ (підпис)

Завдання отримав _____ (підпис)

Анотація

Робота є проектом компанії Genesis, задачею роботи, є створення моделі рекомендаційної системи для оптимізації bid-ставки. Теоретична частина присвячена основним відомостям щодо моделей, які будуть використовуватися, а саме, модель дерева прийняття рішень, випадкового лісу та перехресної перевірки, також наведені основні положення роботи з Apple Search Ads. Ключовими словами є: дерево прийняття рішень, випадковий ліс, перехресна перевірка, Apple Search Ads, cpt-bid, ROI. Для вирішення задачі використовувався пакет rpart у програмі Rstudio. Отримані результати показали, що найкращий результат буде досягатися при дереві рішень, оскільки випадковий ліс містить проблему переобладнання, а перехресна перевірка - недообладнання.

Вступ

З самого початку варто вказати, що ця дипломна робота виконується в межах проекту компанії Genesis, від якої було отримано замовлення, а також яка надала усі необхідні дані для роботи. Але варто відразу вказати, що через відсутність зв'язку з компанією після початку війни, було унеможливлено проведення деяких консультації з представниками компанії, а також можливо не була надана вся наявна у них інформація щодо проекту, наприклад пов'язана з формуванням доходу від реклами для аналізу ROI, який буде описано пізніше.

Recommendation system моделі є важливою частиною нинішньої маркетингової бюджетної політики багатьох компаній, оскільки за її допомогою можливо зробити висновок щодо грамотної стратегії перерозподілу коштів виділених на реклами, тим самим збільшити її ефективність використовуючи модель для обрання найбільш відповідної групи споживачів.

Метою саме цієї роботи є створення Recommendation system моделі для мобільного додатку, який має рекламуватися на платформі Apple Ads Search. В основі роботи покладено управління лише параметром bid-ставки для створення прийнятної моделі, яка б забезпечувала позитивний ROI-показник, що означав би ефективний розподіл рекламного бюджету. Тобто, за допомогою даних за минулі періоди, робиться прогноз того, яку ставку використовувати для максимізації ефективної рекламної компанії. Аналіз відбувається за допомогою моделі дерева, яка надалі покращувалась би моделями випадкових дерев та перехресної перевірки.

Новизною роботи, є створення ефективного алгоритму, який забезпечує прогнозування ставки біду при аукціоні другої біда. Тобто він може встановлювати таку ставку біду, при якій модель стає roi-позитивною,

що забезпечує оптимальний бюджет для програм, які купують рекламу у Apple Search Ads.

Робота складається з двох розділів.

Перший розділ складається з теоретичної інформації, яка описує з роботою алгоритмів дерева, перехресної перевірки та випадкового лісу. Також у ньому описується робота Apple Ads Search та алгоритм, який використовується цією системою у тому числі, пояснена робота ставки другого біда та розрахунок основних маркетингових понять, у тому числі ROI та ROAS.

Другий розділ складається містить опис бази даних (dataset) з практичної реалізації дерева на базі даних, які були надані компанією-замовником. Представлено реалізація алгоритму дерева, випадкового лісу та перехресної перевірки на вказаному dataset. А також за допомогою найбільш оптимальної моделі було надано прогноз cpt bid для ROI негативних даних.

РОЗДІЛ 1. ТЕОРЕТИЧНА СКЛАДОВА ДЕРЕВ ТА МОДЕЛІ

Частина 1. Дерева

Одним із способів вирішення задачі, з прогнозування даних є використання дерев рішень. Щодо визначення, то їх називають таке дерево рішень, у якого внутрішні вузли використовуються для перевірки вхідних даних з загальної множини даних. Варто сказати, що дерева використовуються для різних цілей. Але в основному у них 3 задачі. По-перше, це опис даних, оскільки дерева можуть компактно зберігати інформацію. По-друге, класифікація інформації, але змінна повинна бути дискретною. По-третє, дерево рішень встановлює залежність цільової змінної від вхідних даних для прогнозування.

У даній роботі, дерево рішень буде використовуватися саме для прогнозування, таке дерево називається, регресійним деревом рішень. Структура дерева поділяється на 3 основних елементи. Перше, це коренева нода (позначається у вигляді квадрата), яка є першим розділенням. Друге, це внутрішні вузли (у формі прямокутників), представляють собою умови тестування. Третє – це листкові ноди (у вигляді овалів), вони виступають у вигляді вже остаточних прогнозів. Також варто наголосити на тому, що листкові ноди можуть набувати характеристики чистих листків, це відбувається у тому випадку, коли вони містять у своєму складі дані, які відносяться до тої чи іншої цільової змінної [1].

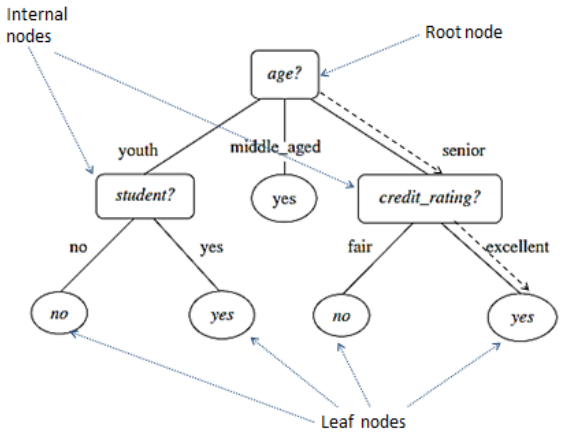

Рисунок 1.1. – Приклад дерева рішень

Оскільки регресійні дерева рішень це певний механізм, то для його роботи необхідно користуватися алгоритмами.

Розглянемо наступні алгоритми.

Першим є алгоритм Ханта. В його основу покладено розбиття даних на чистіші підмножини. Всього він приймає три вхідних значення. Перше – початковий dataset (D). Друге – атрибути (Att), тобто фактор в датасеті і критерії тестування, які як раз дозволяють встановити розподіл атрибуту. Наприклад в ‘quantity>30’, quantity – атрибут, >30 – його критерій. І останнє це “attribute_selection_method” який необхідних для того, щоб визначити ту процедуру, яка найкраще підійшла б для сортування даних по нодам. Щодо самого алгоритму, то він складається з 5 основних кроків [5].

- створення ноди N.
- приведення D_t до класу y_t (листкова нода).
- перевірити, якщо D_t не порожній набором, а Att - порожній, тоді індекс t це листова нода, помічена, як більшість записів.
- перевірка, чи D_t належить більше ніж одному класу, і якщо відповідь позитивна, то необхідно використати згаданий вище «Attribute_selection_method»
- повторення кроків 2,3,4 за необхідністю.

Другий найпоширеніший алгоритм – CART (Classification and Regression tree). В основному, користуються цим алгоритмом у тих випадках, коли необхідно побудувати бінарне дерево. Алгоритм до того ж має механізм, за допомогою якого можна відсікати дерево, алгоритм, що допомагає впоратися з проблемою пропущених значень, а також він використовує індекс GINI, про який буде сказано нижче, а також, він має високу швидкість побудови.

Останній алгоритм, який варто описати – алгоритм C4.5. В його основі покладено побудову дерева, яке має необмежену кількість гілок біля вузла. Основне, для чого було розроблено цей алгоритм – це побудова класифікаційних дерев, оскільки він може приймати лише дискретні значення. Але цей алгоритм має також власні проблеми, через необмеженість гілок, він повільно працює з великими базами даних [6].

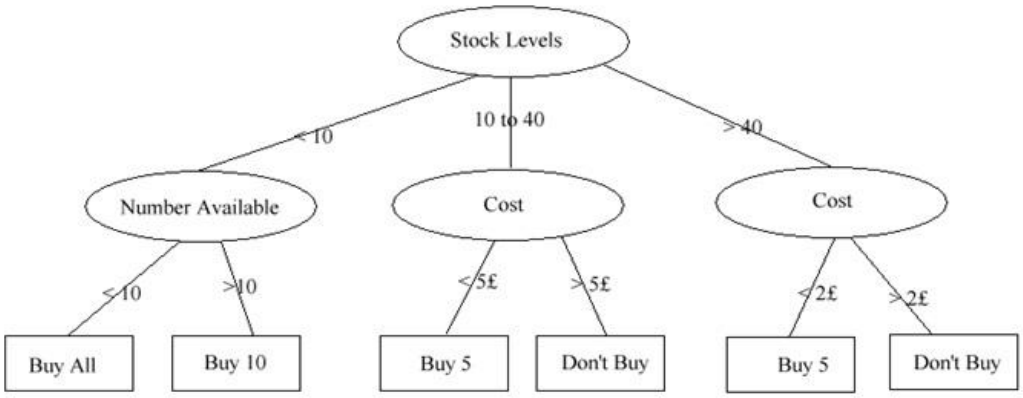

Рисунок 1.3. – приклад алгоритму C4.5.

Загалом, всі сучасні алгоритми пов’язані з регресійними деревами рішень мають схожу структуру. По-перше вони всі йдуть по дереву згори донизу. По-друге, вони мають загальну мету, а саме, за допомогою розбивки атрибутів на множини по їх ознаках, згрупувати значення таким чином, щоб у листках, кількість об’єктів які матимуть інші характеристики була мінімальною. Це є однією з проблем, яка виникає з регресійними деревами.

Всього існують три основні проблеми з регресійними деревами. Першою з них є те, проблема вибору тестових умов, тобто, яким чином краще розділити дерево на ноди. В основному є два види розподілу, це бінарний, тобто той, який дозволяє від однієї ноди провести тільки два варіанти розділення, і багатосторонній поділ – той який дозволяє від одної ноди провести будь-яку кількість «гілок» (більше двох). Насправді, за бажанням, або потребою, можна легко перетворювати один вид розділення у інший. Зазвичай це робиться групуванням. Якщо ж датасет складається з неперервних значень, то варто наголосити, що необхідно у розкладі, особливо небінарному, врахувати усі можливі значення. Якщо перша проблема стосувалась саме того, скільки має бути гілок, то друга, це те, яким чином розділяти значення всередині них (тобто те, як визначити найкращу умову розділення) [2].

Найпоширенішими є 3 метода вимірювання. Перше – ентропія (E(x)), яка вимірює ступінь невизначеності, використовуючи формулу:

$$E(x) = \sum_{i=1}^{n} p_i * log_2(p_i)$$

де p – ймовірність.

Наступним є індекс GINI, який вимірює ймовірність того, що змінна буде неправильно класифікована. Визначається формулою:

$$GINI(x) = 1 - \sum_{i=1}^{n} p_i^2$$

І останній варіант виміру – це класифікаційна помилка. За її допомогою вимірюються неправильна класифікація? Загалом вони відрізняються сферами використання. Якщо індекс GINI використовується в основному в

класифікаційні-регресійні деревах (CART), інші два необхідні в основному для ID3 та деревогенеруючих алгоритмів [6].

Третьою проблемою дерев є переобладнання або недообладання даних, це виникає в основному через те, що зазвичай немає можливості перевірити результат на подібних вибірках. Для вирішення цієї проблеми зазвичай використовують більш складні алгоритми, які включають в себе розбиття даних або інші подібні процедури, наприклад, використання перехресної перевірки або лісу.

Варто також торкнутися проблеми редукції дерев - видалення піддерев, що мають недостатню статистичну надійність. Редукція використовується у тому випадку, коли необхідно видалити ті чи інші піддерева. Робиться це з метою зменшення дисперсії, оскільки деякі піддерева мають занадто низьку статистичну надійність. Навчальна вибірка певним чином постраждає від подібних дій, але вже на тестовій вибірці це сильно допоможе у покращені ефективності.

Загалом, на практиці застосовують 2 найпростіших види редукції. По-перше, це передукція. В основу методу покладено зупинку росту дерева, ще на етапі його побудови. Критерієм зупинки створення нової гілки може виступати певний предикат, який, якщо не буде дотягувати до порогового значення, до дерево не буде розгалужуватися у тому напрямку. Цей метод не визнається як найефективніший спосіб уникнути переобладнання моделі, через саму суть жадібного алгоритму створення дерева, який і використовується у моделі. По-друге, це постредукція. На відміну від передукції вона аналізує вже створені внутрішні вершини дерева на предмет їх недостатньої статистичної надійності. Основним критерієм відбору того, яку гілку варто обрізати є зменшення кількості помилок не на навчальній вибірці, а на тій, яка була виділена з неї до цього і на якій проводиться тестування основного результату. Зазвичай для тестової вибірки виділяють 30 відсотків від усієї бази даних. Спочатку, береться контрольна вибірка, яку

пропускають через повністю побудоване дерево. Далі за допомогою мажоритарного правила, яке в своїй основі має класифікацію піддерев за дочірніми вершинами, отримується результат. Цим результатом можуть стати такі варіанти як, збереження піддерева, заміна у піддерева лівої чи правої дочірньої вершини або повна заміна всього піддерева [1].

Для роботи з деревами в R, перш за все необхідно встановити бібліотеку **rpart** та **rpart.plot**. Для генерування дерев використовують функцію **rpart**(формула, датасет, метод генерування).

Частина 2. Перехресна перевірка

Перехресна перевірка (cross-validation) моделі необхідна для того, щоб зрозуміти яка модель краще працює не на тестовій вибірці, де вона через перенавантаження може показувати гарні результати, а в умовах роботи з тестовим database. Це важливо, особливо при роботі з великими базами даних, оскільки при використанні виключно тестової вибірки неможливо побачити реальну картину, і зрозуміти, які дані будуть зайвими. Зазвичай ця перевірка використовується для регресійних дерев та лісу. Також варто відмітити, що перехресна перевірка добре справляється із ситуаціями, коли отримання додаткових перевірочних даних є неможливим.

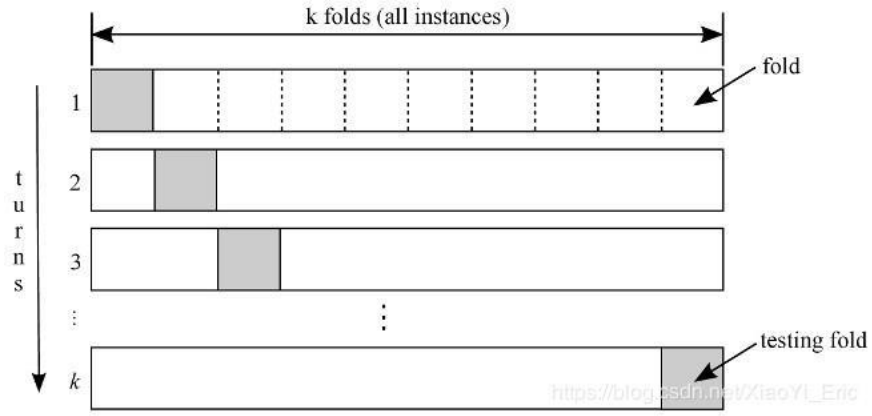

Рисунок 1.4. – принцип роботи перехресної перевірки

Загалом виділяються два методи перехресної перевірки. Перший – працює виключно з однією моделлю і маніпулює даними всередині неї. Другий – потрібен, якщо необхідно порівняти кілька моделей.

Отже, якщо казати, про метод перевірки через одну модель, то в його основу покладено, розбиття вибірки та її перемішування. Всі дані в базі даних розділяються на дві загальні частини, перша це навчальна множина, яка складається з елементів dataset та використовується для того, щоб модель можна було проаналізувати, та набір перевірки, який необхідний для перевірки моделі [5].

Всього є два основних методи перехресної перевірки.

Перший – неповна перехресна перевірка ще називається – k-fold, його суттю є використання декілька способів поділу вихідної вибірки, тобто є можливість вибирати, на скільки частин і як саме будуть поділятися дані.

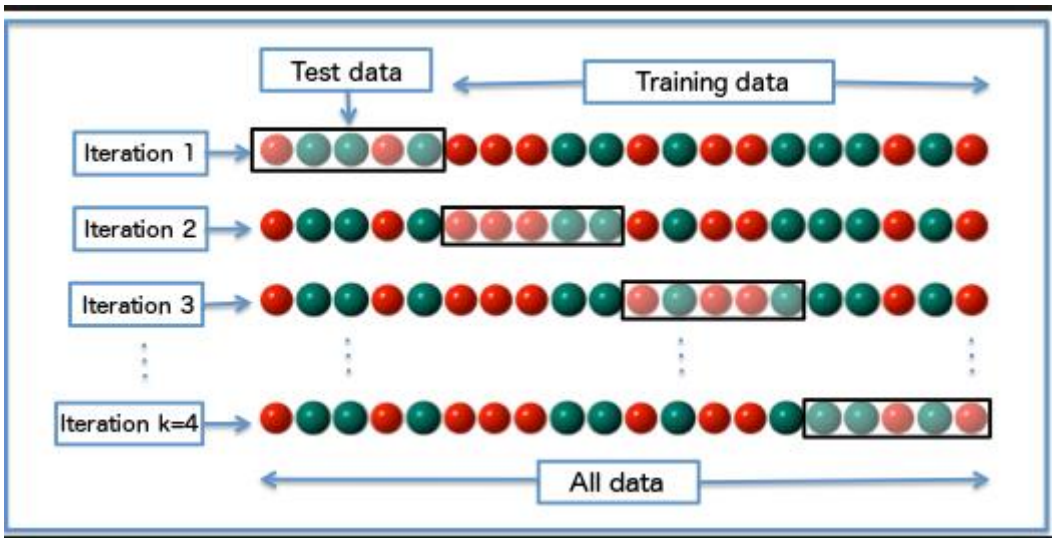

Рисунок 1.2. – k-fold, де кулі різних кольорів, це різні види даних

Другий – повна перехресна перевірка LOOCV, вона у свою чергу, вже включає всі можливі варіанти розділення вибірки. Цей метод є надійнішим, ніж класична неповна перехресна перевірка, але він є дорогим з точки зору обчислювальних ресурсів через проведення більшої кількості операцій, тож його використання небажане у масивних вибірках. Також у повній

перехресній перевірці часто зустрічається проблема переобладнання, через використання всіх даних у процесі навчання [5].

Крім цього є кілька методів, які популярні для виконання певних задач, найпоширенішими з них є такі:

Метод утримання – в його основі є розбиття всієї множини (dataset) на дві рівні частини, одна з них використовується для навчання, друга для тестування, до того весь процес виконується лише один раз. Його перевагами є швидкість та ефективна робота з великими базами даних, але через невелику кількість розбиття, цей метод видає достатньо велику похибку.

Другим є метод повторюваної випадкової підвибірки. Він є дуже схожим на метод k-fold, який був розглянутий вище, можна сказати, що є його різновидом. Основна його відмінність від класичного методу, це те, що у ньому кожен тестовий набір вибирається випадково, що дає можливість для більшої рандомізації, а отже і меншої похибки фінального результату.

Якщо говорити про загальний алгоритм перехресної перевірки, то він складається з трьох основних пунктів. А саме:

По-перше, якщо говорити про метод перевірки (з можливістю самостійного вибору розподілу), то необхідно вибрати кількість розподілів (К). Зазвичай на практиці К в межах від 5 до 10, оскільки, при занадто великій кількості розподілів, вийде перенавантажена модель, оскільки, забагато значень будуть у складі навчальної вибірки, а при замалій кількості, навпаки, можливо будуть пропущені деякі залежності. Інколи виникають ситуації, коли для деяких моделей кількість рекомендованих розділень буде більшою 10 або меншою 5.

По-друге, необхідно використати $K-1$ розділень для того щоб утворити навчальну множину і інші K, які залишилися, використати, як набір для перевірки, тобто, як тестову множину. Наступним кроком, треба протестувати навчальну множину та перевірити її на тестовій, і тоді записати

похибку та загальну точність. Похибкою в даному випадку є сума різниці даних, які були отримані в результаті прогнозування від даних датасету, у якому ці значення присутні.

По-третє, необхідно повторити аналогічні кроки для всіх тестових вибірок.

Останнім пунктом є розрахунок середнього значення точності по усім даним.

$$\bar{T} = \frac{T_1 + T_2 + \dots T_n}{n}$$

де $\overline{T}$ – середня точність, n – кількість ітерацій, T_n – точність ітерації n

Також варто наголосити, що якщо відхилення від середніх значень тестової множини є достатньо великим, то це означає, що, або вибірка є досить малою, або модель сильно залежить від конкретних блоків значень [5].

Щодо порівняння кількох моделей, то там працює інша ідея, а саме, порівняння моделей, які проаналізовані різними алгоритмами та підходами, на основі однієї навчальної вибірки. Але цей спосіб має і свої недоліки, і основний з них, це те, що інколи неможливо визначити, який з алгоритмів працює найкраще, виникає це по тій причині, що точність кожного алгоритму перевіряється по трьох вибірках, це тренувальна, перевірочна та тестова. А отже, часто виникають ситуації, коли для деяких вибірок значення, утворені в результаті прогнозування більші за фактичні серед всіх алгоритмів, а деякі менші. Тож цей вид порівняння найчастіше застосовують для виявлення проблем алгоритму, який використовують [5].

Щодо алгоритму порівняння, то:

По-перше, необхідно обрати з якими даними будуть робитися маніпуляції, розбити їх по трьох вибірках, а саме виділити тренувальну, перевірочну та тестову вибірки та обчислити точність кожної із них.

По-друге, порівняти дані за допомогою графіка або таблиці, куди занести дані про результати обчислення точності, та проаналізувати їх.

Наостанок варто наголосити, що необхідно правильно користуватися моделями перехресної перевірки і розуміти яка у них загальна функція. Часто зустрічаються ситуації, коли структура самої моделі змінюється в результаті чого набори даних для перевірки та навчальні дані можуть мати суттєві відмінності. До того ж необхідно відмітити, що перехресна перевірка є затратною у вартості обчислення, оскільки необхідно проводити не одну операцію з моделлю, а стільки операцій, на скільки була з самого початку розбита база даних. Але проблему затратності можна вирішити, зазвичай це робиться шляхом попереднього обчислення значень, які вже використовувалися з тренувальними даними. Проте, незважаючи на те, що подібні методи можуть допомогти у ситуації з занадто великими затратами у обчисленні, їх необхідно використовувати обережно, повністю відділяючи тренувальний та навчальний набори. У іншому випадку, виникне інша проблема, а саме проблема зміщеної оцінки набору даних [5].

Крім помилки, яка включає в себе змішування даних навчальної і тренувальної вибірки, варто виділити ще такі рекомендації які, допомагають ефективно використовувати cross-validation та мінімізувати похибку.

По-перше, при використанні перехресної перевірки на кількох ітераціях необхідно брати результати з усіх, а не тільки з найефективнішої.

По-друге, помилкою також буде використання інформативного набору параметрів, який базується на проведення початкового аналізу даних усього набору даних. Інформативним набором параметрів є такі параметри, які є корельованими до набору даних, який необхідно спрогнозувати. Оскільки це веде до спотворення результату перехресної перевірки.

Щодо переваг класичної перехресної перевірки, як методу, то варто виділити такі аспекти.

По-перше, через рандомізоване перемішування даних при їх кожній розбивці на навчальну та тренувальну, зводиться до мінімуму ймовірність того, що в результаті перевірки моделі, середня помилка буде значною.

По-друге, у порівняні, наприклад, з методом перевірочної вибірки, де розбиття на дві групи відбувається методом звичайного розділення даних навпіл, у перехресній перевірці, зазвичай база даних розподіляється на 5-10 частин, де одна з них, це навчальна частина, а інші тренувальні. Це дає інформацію щодо чутливості моделі до вибору даних з навчального набору [5].

Якщо говорити про використання перехресної перевірки у R, то вона проводиться 4 основними функціями з таких пакетів: **Cvlm** з пакету **daag**, **cvlm** з пакету **cvTools**, **cv.glm** з **boot** та **train** з **caret**. У даній роботі буде використовуватися пакет **caret**.

Для роботи функції **train** необхідно задати наступні аргументи. Далі буде наведено найважливіші з них:

- **Method**. Метод за яким мають створюватися ітераційні вибірки.
- **Number**. Цей параметр задає кількість ітерацій при повторних вибірках. Як вже зазначалося раніше, зазвичай береться число від 5 до 10.
- **Repeats**. Аргумент який дає можливість вибирати кількість повторень для виконання перевірки.
- **P**. Наступний критерій визначає те, якою має бути частка навчальної вибірки від загальної кількості даних.
- **Search**. За допомогою цього параметра можна налаштувати те, як будуть перебиратися вибірки. Зазвичай використовують два типи. Тип “grid” - перебір відбувається по наперед заданій сітці та “random” при випадковому переборі.

Частина 3. Випадковий ліс.

Випадковий ліс (Random Forest) – метод машинного навчання, який допомагає покращити результати регресійних дерева. Він був розроблений ще в попередньому сторіччі Лео Брейманом та Адель Картер. У своїй основі він представляє собою архітектуру, яка називається ансамблевою. З моменту свого створення він майже ніяк не змінилася і досі є популярною для вирішення широкого спектру задач. Серед її основних переваг є такі. По-перше, цей метод гарно підходить для більшості задач класифікації, тобто показує себе ефективно для більшості моделей. По-друге, варіації Random Forest (надалі RF) також гарно підходять під задачі не тільки класифікації, а і селекції ознак, кластеризації, регресії та пошуку аномалій.

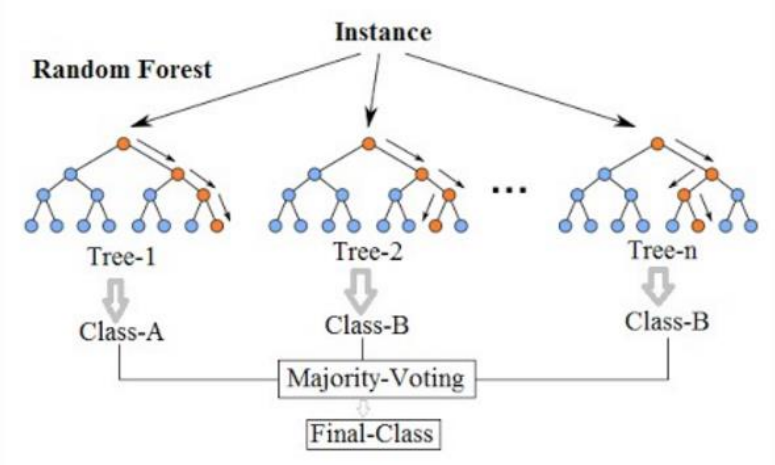

Рисунок 3.1. – принцип роботи випадкового лісу. Кулі синіх кольорів, це загальний розподіл дерева, жовті, те, яким чином йти по дереву для досягнення відповідного резальтату

В управлінні підприємствами та у бізнесі загалом, концепцію випадкового лісу часто використовують для так званих моделей «чорних скриньок», через те, що вона, як вже описано вище, має широкий спектр використання, до того ж він не вимагає значної конфігурації [9].

Як і більшість подібних розширень моделі дерев, наприклад, перехресна перевірка, в основу Random Forest покладено порівняння різних варіантів

Tree моделей між собою для виявлення помилок, зменшення дисперсії та усунення зміщення. Основним фактором через що модель з багатьма деревами працює краще, ніж модель з окремим деревом, це так званий ефект «мудрості натовпу», він говорить про те, що при достатній кількості слабо корельованих даних, їх загальна дисперсія буде низькою, а зміщення мінімальним (прикладом такого ефекту є інвестиційний портфель, який є надійним, якщо в ньому зібрані низькокорельвані цінні папери).

Низьку корельованість між окремими деревами ліс гарантує двома основними методами. Перший метод має назву **bagging**. Його особливістю є рандомізація дерев у лісі. Тобто за її допомогою, відбувається вибір кожним окремим деревом випадкових даних, в результаті чого, кожне наступне дерево буде відрізнятися від попереднього. До того ж вибірка відбувається з відшкодуванням, тобто методом bootstrapping, він дає можливість використовувати дані повторно для інших дерев. Тож метод дає можливість збільшити варіативність лісу не жертвуючи загальною точністю. Другий метод має назву «випадковість ознаки». В його основу покладено принцип розділення вузла кожен раз по новим атрибутом. Тобто, якщо в **bagging** рандомізація досягалась шляхом випадкового підбору даних для кожного окремого дерева, то у випадку другого методу, рандомізуються самі атрибути з підмножини вибраної до цього вибірки атрибутів. Цей метод у поєднанні з першим створює ще більшу рандомізацію, а отже і зводить кореляцію між окремими деревами до мінімуму [9].

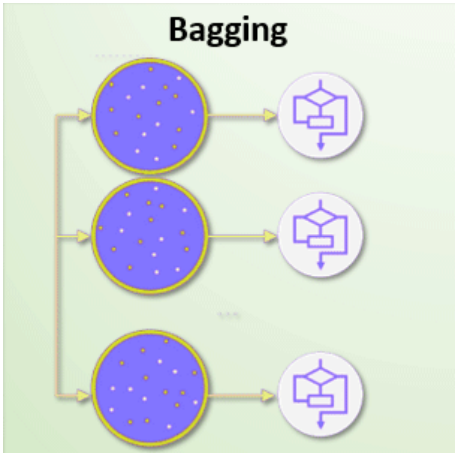

Рисунок 3.2. – принцип роботи bagging. Сині кола, умовна демонстрація рандомізації даних.

Щодо механізму лісу, то він включає в себе створення N кількості дерев. Тобто очевидно, що модель, як вже сказано раніше, відповідає принципу ансамблювання, цей принцип включає в себе створення певної кількості окремих дерев, для того, щоб потім їх поєднати і покращити результат Decision Tree. Ці дерева створюються випадково, тобто за основу береться не те, наскільки велика кількість значень того чи іншого атрибуту відноситься до певного класу, а в основі лежить рандомізація. Серед всіх дерев, які були отримані описаним механізмом, в результаті певної процедури голосування вибираються ті, які найбільш точно відповідають заданій базі даних, а також вирішується проблема зміщення окремої моделі дерева.

Взагалі всі дерева в лісі будується по такому механізму:

- Вибирається певна підвибірка атрибутів з основної вибірки. Число значень атрибутів нехай позначається як k, а загальні значення в вибірці – m. Зазвичай $k = \sqrt{m}$.
- В якості найкращої точки розділення серед усіх перед цим вибраних k за допомогою GINI рахується деяка нода D.
- Використовуючи метод поділу, який обраний як найефективніший необхідно розділити ноду D на інші вузли.
- Необхідно повторити всі попередні кроки для побудови одного дерева.
- Повторити всі кроки n разів для того, щоб створити випадковий ліс з n кількістю дерев [5].

Після побудови лісу з n дерев, необхідно проаналізувати знайдені результати, для цього необхідно зробити такі кроки:

- По-перше, необхідно взяти кожне дерево створене за відповідним методом та знайти прогноз

- По-друге, необхідно зберегти прогнозовані цільові результати усіх N дерев, для подальших розрахунків
- По-третє, треба підрахувати кількість співпадінь за кожен цільовий результат усіх N дерев
- Вибрати серед усіх цільових результатів, той, у якого більше всього співпадінь. Він і буде фінальним прогнозом

Загалом на практиці виділяють такі особливості випадкового лісу. По-перше, як і у випадку з деревами, максимальна кількість атрибутів, які будуть використовуватися у моделі, не приведе до того, що модель буде найкращою. По-друге, так само зберігається і проблема дерев у тому, що на наборі навчальних даних, результат у більшості випадків краще, ніж на наборі тестових даних, це означає, що висока точність на навчальній вибірці не може гарантувати таку ж саму, або хоча б схожу точність і на тестовій, тож використання перехресної перевірки бажане не тільки для окремого дерева, а і для всього лісу. Ці дві проблеми означають, що модель схильна до проблеми переобладнання. Сама проблема може виникати через деякі причини, основними серед них є набір даних, які можуть бути погано рандомізованими, велика кількість атрибутів та початковими параметрами моделі. Також варто зазначити, що при обрані мінімальної кількості атрибутів буде виникати проблема недообладнання, в результаті чого, модель теж буде неефективною [5].

Якщо казати про недоліки моделі, то можна виділити ще два. По-перше, занадто велика модель, яка формується в результаті аналізу моделі. По-друге, при великій кількості даних, побудова лісу стає досить складною задачою та вона займає багато часу, через велику кількість дерев, які необхідно створювати для генерування лісу. Попри те, що ліс містить подібні проблеми, що містили окремі дерева, все ж на практиці, точність лісу завжди буде вища ніж окремого дерева, якщо вони використовуються для аналізу

однієї й тої самої бази даних, та якщо кількість і якість атрибутів буде однакова для обох видів моделі.

Щодо переваг моделі, то у першу чергу варто виділити саме такі плюси. По-перше, подібний тип моделей добре підходить, якщо база даних містить багато атрибутів та класів. По-друге, моделі є ефективними як при неперервних, так і при дискретних атрибутах. По-третє, важливою перевагою є невисока чутливість до монотонних перетворень, у першу чергу до масштабування.

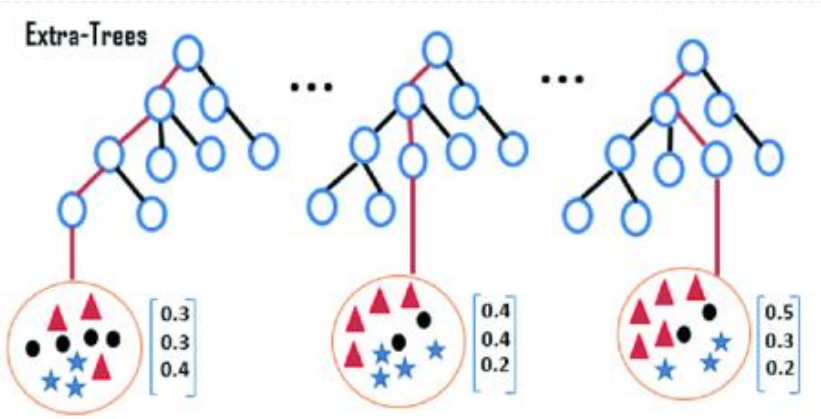

Рисунок 3.3. Extra trees. У кругах знизу знаходяться умовні результати дерева, де вбік від них показані точності на трьох видах даних.

Останнє на чому варто наголосити, це те що в моделі існують методи оцінювання значимості окремих ознак моделі. Для покращення рандомізації випадкового лісу, а отже і збільшення його ефективності можна також використовувати додатковий крок рандомізації, який називається ExtraTrees. Незважаючи на те, що алгоритм дії надзвичайно рандомізованих дерев доволі схожий зі звичайним алгоритмом лісу, він має 2-ві основні відмінності від класичного алгоритму. По-перше, у ExtraTrees використовується відразу уся вибірка, у порівнянні з лісом, де використовується підвибірка, що складається з вхідних даних з заміною. Це допомагає зменшити зміщення. По-друге, розгалуження дерев відбувається не за чіткою схемою, тобто найбільш оптимальним розгалуженням, а це відбувається випадковим чином.

Для того, щоб через рандомізацію розгалуження не відбувалося зменшення дисперсії, усі розподіли, які зроблені випадковим чином порівнюються між собою, у межах одного створеного дерева і обирається те розчеплення, яке показує найвищу ефективність. Основною перевагою ExtraTree, крім оптимізації у вигляді зменшення дисперсії і зміщення, виступає те, що цей алгоритм є швидшим, через те, що немає необхідності у пошуку оптимального розгалуження [3].

Для використання моделей RF у R studio, зазвичай використовують пакет під назвою – ‘randomForest`. Основною функцією для створення дерев у ній є функція **RandomForest()**. У ній необхідно задати такі параметри, як

- дані, які будуть використовуватися у моделі випадкового лісу.
- кількість дерев, які будуть всередині лісу, їх діапазон складає значення від 0 до 1000
- кількість випадкових атрибут, як вже зазначалося вище, найкраще брати значення $\sqrt{N}$
- максимальна глибина дерева. Значення від 1 до максимального зростання дерева. Це позначається як -1.
- мінімальна кількість кінечних вузлів. Приймаються значення від 2 до нескінченності.

Частина 4. Apple ads search

Реклама продукту є напевно не менш важливою частиною бізнесу, ніж саме виробництво, оскільки за її допомогою фірма збільшує попит на свої товари, а отже збільшуються як продажі, так і в результаті безпосередньо виробництво. На сьогоднішній день, велика частина реклами перебуває в інтернет просторі, особливо та, яка стосується інноваційних продуктів. Наразі одним з основних майданчиків для розміщення реклами в мережі виступає apple ads search. По кількості реклами він займає третє місце, після Google ads та facebook ads. Незважаючи на це, за статистикою, рекламний

майданчик від apple займає перше місце серед всіх основних рекламних підрозділів компаній за ефективністю реклами, тобто цей сервіс забезпечує найкращі показники у кількості скачувань на один переглянутий рекламний банер [7].

Загалом, apple ads search (ASA) це рекламний інструмент, який використовується у Appstore та відноситься до класу утиліт, які просувають рекламу по ключовим словам, а не по схемі PPC, яка початково була основною у цьому сервісі та в основу якої покладено оплату за один клік. В той же час принцип просування за ключовими словами означає, що при вводі певної сукупності слів у стрічку пошуку, внутрішній алгоритм визначає потребу людини і виводить рекламний банер певної компанії за принципом аукціону другого біда, його розглянемо нижче. У ASA є можливість вибирати не тільки одне слово і не словосполучення, а також створювати групу оголошень за найефективнішими ключовими словами, що і допомагає збільшити конверсію [7]. Конверсія – це відношення числа переглядів реклами до числа людей, який зайшли у додаток.

$$K = \frac{W}{V}$$

де K – коефіцієнт конверсії, W – кількість переглядів, V – кількість відвідувачів сайту

Якщо говорити про переваги ASA, то в першу чергу необхідно звернути увагу на 7-м основних пунктів.

По-перше, ASA дає більш детальну статистику, у порівнянні з іншими утилітами, що пов’язана з конверсією рекламних запитів. Тобто вона дає можливість продивитися країну, регіон, вік, стать та інші характеристики покупця. Тим самим цю інформацію можна використовувати не тільки для зміни деяких налаштувань та App store optimization (ASO), а і на інших рекламних майданчиках. До того ж, ASA дає можливість сегментувати

власну аудиторію, оскільки детальна статистика дає розуміння, які користувачі вперше встановили додаток, вже бачили, але не встановлювали утиліту, або взагалі не бачили рекламний банер, тож на основі цього є можливість розробки якісного маркетингового плану без залучення додаткових компаній.

По-друге, у ASA є багато різних розширень, які допомагають оптимізувати маркетингові затрати. Найпопулярнішими серед них є такі, які слугується розширенням ASO, а саме BroadMatch і SearchMatch. BroadMatch використовується для майданчиків, які застосовують пошук за ключовими словами, за його допомогою, алгоритм використовує не тільки одне конкретно введе слово чи словосполучення для видачі рекламного банера, а також видає рекламу для схожих за сенсом або граматикою слів. У свою чергу, SearchMatch застосовуються у випадках, коли необхідно швидко та легко почати показ реклами. За допомогою цієї функції можна виставляти рекламні банери, навіть тоді, коли не були ще підібрані всі ключові слова та не визначені для них bid ставки. Якщо ключові слова ще не були підібрані, то релевантність рекламного банеру та конкретного продукту пошуку клієнта встановлюють за такими параметрами додатку, як, його назва, підзаголовок та опис.

Broad Match	Exact Match	Search Match
Keyword + misspellings, plurals, closely related words, synonyms, related searches, related phrases, and translations	Keyword + common misspellings and plurals	Apple finds relevant keywords and search terms depending on app metadata
Wider Audience	Narrower Audience	Wider Audience
More impressions	Fewer impressions but higher TTRs and CVR	More impressions
Discovery and General Campaign	Brand and Competitor Campaign	Discovery Campaign

Рисунок 4.1. – найпоширеніші розширення ASA та їх короткий опис

По-третє, через свою концепцію а також те, що люди, які користуються Appstore за статистикою більш багаті, рекламні банери працюють більш якісно і релевантно, ніж на інших майданчиках. Якщо на інших майданчиках, в середньому коефіцієнт конверсії буде близько 50%, то у ASA показник коливатиметься між 60-70%. До того ж Apple add search рекламує виключно утиліти, до того ж реклама відбувається тільки у додатку AppStore. Це означає, що потенційний клієнт вже буде більш зацікавленим у рекламованій продукції, оскільки заходячи у AppStore, покупець з самого початку планує встановити якусь програму. Крім цього така висока конверсія досягається тим, що рекламні банери органічно розміщенні серед основного функціоналу, тобто, вони не заважають користувачеві та за своїм дизайном підігнані до дизайну самого додатка, тим самим, у користувача не виникає бажання відразу її пропустити.

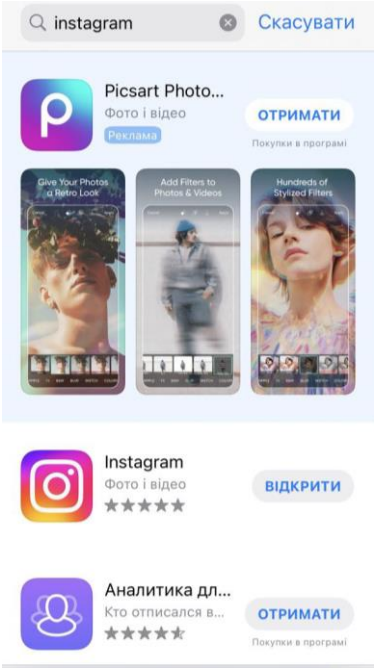

Рисунок 4.2 – Вивід реклами у додатку AppStore

По-четверте, реклама у ASA дає можливість забирати трафік у конкурентів, це відбувається через алгоритми, які працюють з ключовим словам.

По-п’яте, у ASA є можливість захисту власного бренду. Це працює так, що потенційний клієнт, який вводить назву бренду певної компанії бачить рекламний банер саме тої компанії, яку він конкретно шукав, а не їх конкурентів.

По-шосте, ASA має такий позитивний момент, як те, що надає дані по кількості установок навіть у тих користувачів, які відключили трекінг реклами, що дозволяє ще більш точно зібрати статистику по користувачам.

Останнім пунктом виступає вбудована функція копіювання. Вона дозволяє використовувати персональні налаштування ASA не тільки у магазині регіону компанії, але і переносити їх у магазини інших регіонів. Тим самим повністю клонуються групи оголошень, ключові слова, демографічна статистика та навіть наявні суми ставок. Всю скопійовану інформацію можливо відредагувати для ефективного подальшого використання у певному регіоні [7].

Хоча ASA по сукупності переваг має бути очевидним лідером з використання реклами серед своїх конкурентів, але це не відбувається по тим причинам, що вона також має свої недоліки.

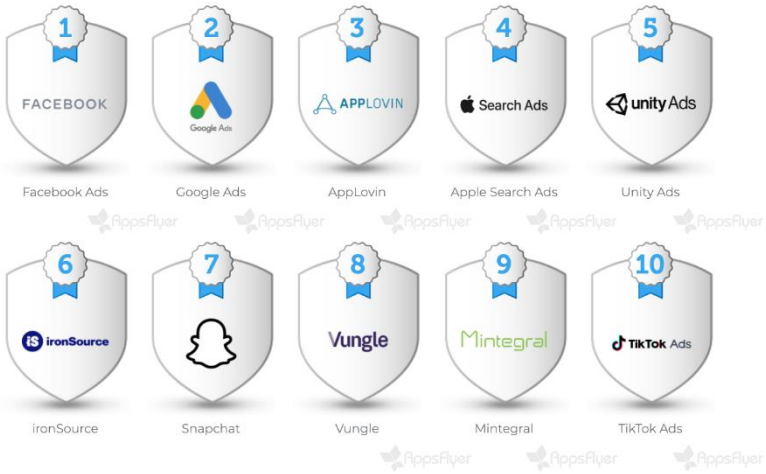

Рисунок 4.3 – рейтинг компаній, які надають рекламні послуги в інтернеті

Першим з них є ціна. Як вже говорилось раніше, коефіцієнт конверсії у ASA найбільший, але це частково компенсується доволі високою ціною

розміщення рекламних банерів у порівнянні з основними конкурентами. По-друге, apple ads search має порівняно менший трафік. Це пов’язано з тим, що рекламу у ASA можуть бачити лише ті потенційні клієнти, які користуються продуктами Apple та до того ж витрачаються багато часу у Appstore. Звісно з одного боку, це позитивно впливає на конверсію, проте, це негативно впливає на загальну кількість переглядів банера. Хоча навіть на здавалось би менший трафік, ніж у конкурентів, все одно близько 2 мільйонів програм використовують саме майданчик AppStore. По-третє, apple ads search має проблему так званої «канібалізації» трафіку. Тобто, виникають ситуації, коли компанія платить за показ реклами, там, де її продукт і так знаходиться на першому місці серед пошукових значень. Проте, ця проблема є загальною для всіх рекламних майданчиків, які використовують за основу пошук за ключовими словами та розподіл реклами відбувається за принципом аукціону, приклади наведемо пізніше [7].

Далі в роботі будуть описані особливості роботи в ASA. В першу чергу, це широкий вибір таргетованість на певні характеристики економічних агентів. Цими характеристиками виступає вік, країна, регіон та інші різні соціально-демографічні показники. Також агент, який має бажання користуватися ASA повинен мати Apple ID для доступу до персонального кабінету.

Крім цього, концепцією оплати за рекламний банер виступає CPT, або за бажанням CPA. CPT – означає, що фірма має платити середню ціну за натискання на рекламний банер, в свою чергу, CPA – дозволяє платити середню ціну саме за встановлення додатку рекламодавця. Безпосередньо оплата має здійснюватися через платіжну картку в особистому кабінеті.

Також, варто наголосити, що Apple ads search ділиться два основних види по внутрішньому функціоналу. Перше, це ASA Basic. В основному він використовується серед розробників програм через те, що цей пакет є мінімалістичним, але робочим інструментом, який не займає багато часу, як для того, щоб розібратися у ньому, так і для того, щоб працювати з ним.

Взагалі пакет Basic має багато схожого з початковою системою ASA, оскільки теж використовує PPC. Особливістю цього пакету є те, що реклама демонструється у верхній частині результатів пошуку.

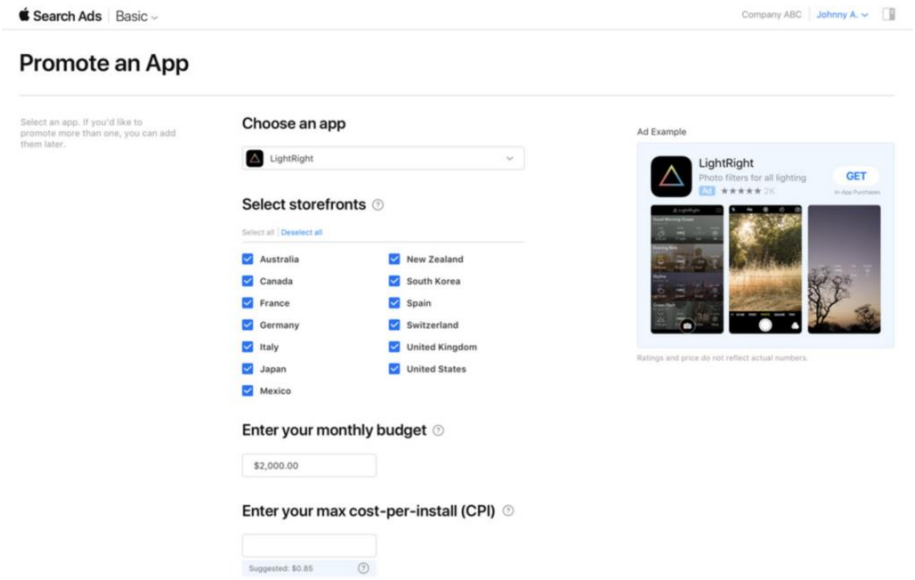

Рисунок 4.4 – вигляд меню ASA Basic

Серед функціоналу, який доступним у Basic, основними є рішення, які дозволяють клієнту не вводити нічого, крім цільових показників після чого алгоритм ASA підбере ключові слова та проведе налаштування потенційної аудиторії, також важливою особливістю є те, що ціна встановлюється клієнтом заздалегідь (у Basic використовують принцип CPA). Але ASA Basic має і свої обмеження. Ними виступає обмежений бюджет у розмірі до 10000 доларів США за один додаток у місяць, а також максимальна кількість утиліт, які ASA зможе рекламувати складає 50.

Другим є ASA Advanced. Його вже зазвичай використовують не розробники, а рекламні агенції, оскільки пересічному розробнику, який не знайомий з інструментарієм apple search ads буде важко розібратися у всіх доступних звітах та налаштуваннях. У порівняні з Basic він показує рекламні банери не тільки в верхній частині результатів пошуку, а і на самій вкладці «пошук», що збільшує кількість потенційних клієнтів. Особливостями Advanced версії виступають такі рішення як, доступ до цілої низки

інструментів маркетингу. За їх допомогою можливе точкове налаштування побажань рекламодавців та можливість вибирати цільову аудиторію, якій буде демонструватися продукт, також з'являється можливість по встановленню і подальшому редагуванні ключових слів. Крім цього, клієнт може бачити детальний звіт по усім ключовим показникам. У порівнянні з Basic, де ціна встановлюється заздалегідь, у цій версії програми є можливість встановлення максимальної ціни як у CPA так і у CPT, до того ж ціна встановлюється за 1000 показів чи натискань на рекламний банер. Обмеження версії Basic, відсутні [6].

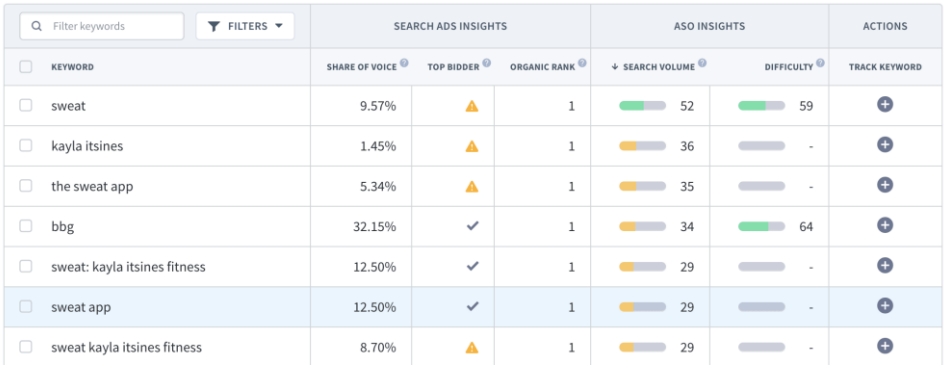

Рисунок 4.5 – демонстрація роботи зі статистикою у ASA Advance

Також у ASA є можливість налаштовувати зовнішній вигляд рекламного поста. У тому разі, якщо клієнт не бажає його змінювати, то формування банера відбувається автоматично, а саме вибирається перше викладене клієнтом відео та три зображення з нього. Якщо ж рекламодавець має бажання змінити візуальну складову, то він може користуватися так званим набором Creative Set. За допомогою якого, з'являється можливість вибирати потрібне клієнту фото або відео [7].

Як вже було сказано раніше, ASA використовує аукціон другого біду, як основний алгоритм вибору демонстрації рекламного банеру тої чи іншої компанії. При цьому варто зазначити, що мається на увазі не класичний аукціон другої ціни, оскільки у випадку ASA існує певна заздалегідь умовлена надбавка. По-перше, варто наголосити, що вищезгаданий аукціон

відноситься до закритого типу, тобто, всі учасники не знають ставку інших. По-друге, **аукціон другого біду**, відрізняється від класичного закритого аукціону (першої ціни) тим, що у ньому перемогу отримує той учасник, який поставив найвищу ставку, але при цьому, він платить суму, яку поставив учасник, який зробив другу найбільшу ставку, при цьому фірма має платити невелику маржу за участь у самому аукціоні, що повинно відобразитися на стратегії, про яку буде написано нижче. [4]

Тож, у цьому випадку, найбільш оптимальну стратегією є робити ставку, яка відповідає граничним доходам фірми з реклами. Оскільки, при цій ситуації, коли фірма спробує поставити більше, інші фірми теж будуть мати спокусу поставити трохи більше за граничний дохід, в результаті чого, ця конкретна компанія отримуватиме збитки і врешті решт піде з ринку. А при ситуації, коли кожен обирає оптимальну стратегію, компанія, яка виграла бід буде у невеликому виграші, а ті, які програли, не отримають збитки. Вигоду від проведення цього типу мають як продавець, так і покупець товару, оскільки учасникам не вигідно підіймати ціну лоту вище, ніж їх граничні витрати, при цьому, ставки все одно вищі, ніж при аукціоні першої ціни.

Аукціон другого біду є поширеним саме у інтернет просторі при торгівлі цифровими товарами або послугами. Це відбувається через те, що, по-перше, він є аукціоном закритого типу, що спрощує його використання в онлайні, до того ж саме аукціон подібного типу проводиться дуже швидко і як вже було сказано, з вигодою для всіх учасників.

Рисунок 4.6 – аукціон другого біда

Варто також зазначити, що аукціон другого біду у ASA відбувається на основі RTB (Real Time Bidding), який дозволяє проводити аукціони між продавцями та покупцями у режимі реального часу. Загалом у цьому типі аукціону покупець робить всього одну ставку, яку вважає оптимальною, що і робить цей аукціон швидшим за його інші види. Звісно подібний тип не позбавлений класичних його проблем, які виникають майже у всіх видах аукціонів. А саме, заниження ціни в результаті змови окремих покупців, якщо вони мають достатньо влади на конкретному аукціоні, також проблемою можуть виступати і самі продавці, які за допомогою алгоритмів, які здатні імітувати ставки штучно їх підіймають. Також цей тип аукціону має певні обмеження.

По-перше, відсутня можливість дослідження ціни, а саме виявлення загальної ринкової ціни за певний товар. Це можна зробити лише у результаті проведення ряду схожих аукціонів.

По-друге, виграш покупців, хоч і є вищим, ніж у аукціоні першого біду, але все одно часто може буде рівним нулю. Тож цей аукціон добре підходить саме для ефективного розподілення ресурсів в цьому випадку, рекламних місць, але не завжди забезпечує потрібний прибуток.

Однак незважаючи на вищеописані мінуси, аукціон другого біду на основі RTB все одно є більш ефективним у разі продажу маркетиногових послуг, особливо у інтернет просторі [4].

Варто також відзначити, що у роботі використовуватиметься ROI-позитивна модель. Це означає, що має бути ROI>ε. Взагалі у маркетинговій науці всього є 3 основних коефіцієнтів, за допомогою яких можливо порахувати збитки та надходження від маркетингових рішень. Це ROMI ROAS та ROI. В

основному ці три коефіцієнти відрізняються між собою тим, які витрати та доходи включаються в конкретний коефіцієнт.

ROI (return of the investment) – коефіцієнт в основу якого покладено розрахунок коштів, які забезпечують повернення вкладених інвестицій.

Він включає в себе всі можливі витрати, а також маржу. За допомогою цього коефіцієнта фірма може зробити висновки щодо рівня ефективності рекламної кампанії, того, яким чином вона вплинула на фінальний результат роботи маркетингового відділу, а також, правильність дій для оптимізації рекламного бюджету та на цій основі створення більш ефективного бюджету на наступний операційний період.

Формула для розрахунку ROI виглядає так:

$$ROI = \frac{I * M - C}{C} * 100\%$$

де I – дохід від реклами за поточний період,

M – маржинальність,

C – затрати, тобто сума грошей, яка витрачається на отримання доходу за певний період.

$$\mathrm{M} = \frac{(P - \mathrm{CC})}{P} * 100\%$$

де P – ціна, CC – собівартість.

При розрахунках ROI важливо не плутати його з подібними вже вищезгаданими коефіцієнтами. Основне чим відрізняється ROI та ROAS це те що, при розрахунку першого враховуються всі затрати в тому числі маржа. При цьому, ROI відрізняється від ROMI тим що ROI рахує рентабельність бізнеса в цілому, у той же час ROMI використовується для розрахунків рентабельності маркетингових інвестицій.

До того ж у деяких ситуаціях ROI є малоефективним інструментом. Це стосується ситуацій, коли необхідно оцінити значення для дорогих товарів та послуг, або коли клієнт думає, чи користуватися товаром достатньо довгий час. Також, коефіцієнт є статичним тож погано працює у довгостроковому періоді, бо унеможливлює імплементування даних пов'язаних зі зміною у ринковій кон'юнктурі. Також, ROI ніяким чином не враховує неекономічні фактори, такі як лояльність до компанії, її репутацію, або висвітлення у засобах масової інформації.

Усі вищеописані коефіцієнти, а особливо ROI необхідно рахувати, оскільки кількісні зміни, які візуально можна побачити в кінці операційного періоду не передають повну картину, та ніяким чином не оцінюють якісну складову маркетингової стратегії, що унеможливлює проведення висновків щодо її загальної ефективності та є причиною пропуску серйозних помилок [10].

Розділ 2. Створення Recommendation System модель у R

Частина 1. Реалізація алгоритму дерева

Для того, щоб почати реалізовувати сам алгоритм, необхідно для початку привести базу даних, у відповідний вигляд. Код у Rstudio приведений у додатку А. Сама база даних представляє собою дві таблиці asa_dataset_final_11112021.csv та roi_by_country_11112021.csv. У ній міститься основна інформація про завантаження по ключовим словам, які використовувалися у певних країнах або регіонах.

Всього представлено два dataset

Перший має назву asa_dataset_final_11112021 і складається з:

date – дата

app_id - id додатку,

campaign_id - назва кампанії,

country_or_region - країна або регіон,

ad_group_id - id групи реклами,

ad_group_name - назва рекламної групи,

keyword - ключове слово,

keyword_id - id ключового слова,

cpt_bid - ставка біда,

keyword_match_type - вид ключового слова (використовується конкретне слово чи підбирається приблизне значення),

keyword_status - статус ключового слова,

keyword_display_status - та статус відображення ключового слова,

spend - витрати,

taps - кількість натискань,

installs - кількість завантажень,

impressions - загальний об’єм реклами на сайті,

ttr - коефіцієнт конверсії

cr - коефіцієнт клікабельності,

avg_cpa - середня вартість реклами за годину

avg_cpt - плата рекламній мережі за дій клієнтів сайту,

new_downloads - нові скачування

redownloads – заново скачані продукти

lat_on_installs - користувач включив LAT.

lat_off_installs – користувач вимкнув LAT

Другий датасет roi_by_country_11112021.csv складається з 3 полів, а саме:

date – дата

country_code – країна або регіон

roi_6m_percentage – roi за відповідну дату та країну

По-перше, необхідно було налаштувати дані в таблиці. Для цього, використовувалися виключно функції в Rstudio, оскільки при розпаковуванні даних у самому Excel, виникали помилки пов’язані з форматуванням. Тож з самого початку, через функцію **read.csv** було імпортовано дані в R.

Наступним кроком, було створено окремі датасети з імпортованих даних, для маніпуляції з їх кількістю стовпців. Також, наступним кроком, було відфільтровано дані від тих, які мають невисокий прогнозний потенціал, а саме, від значень, де, витрати та скачування дорівнювали нулю, а також за

рекомендацією Genesis, було прибрано всі дані, які містили AUTO у полі *keyword_match_type*. Це дозволило скоротити датасет без сильної втрати у точності з 253840 значень до 25796.

Далі, через те, що дані у моделі дерева неперервні, операція виконується довго, тож було здійснено оптимізацію, а саме створення нового стовпчика у якому згруповано дані про bid-ставки. А саме, брався проміжок між парними числами та у клітинку записувалось непарне число, що знаходиться між ними.

Далі проводилась робота з другою таблицею, а саме, було створено новий стовпець *roi_roi.* Туди було записано дані про те, чи є ROI представлений у таблиці позитивним, чи негативним. Наступним кроком, ці дані переносилися у створений у початковій таблиці стовпець *rois*. Це робилося через порівняння відповідних дат та країн у двох таблицях.

Далі всі дані розділялися на ROI позитивні та негативні значення. Всього з 25796 значень, позитивних значень було 13298, а негативних, 12498 значень.

Наступний крок - це побудова дерева. Код моделі дерева продемонстрований у додатку Б.

За допомогою функція **rpart** було створено модель дерева, яка складається з функції **near_cpt_bid** та аргументів campaign_id, spend, taps, installs, impressions, ttr, cr, avg_cpa, avg_cpt, new_downloads, redownloads, lat_on_installs, lat_off_installs.

```
#creating the model of decision tree
set.seed(1234)
decisionTree <- rpart(near_cpt_bid ~ campaign_id+spend+taps+installs+
                        impressions+ttr+cr+avg_cpa+avg_cpt+
                        new_downloads+redownloads,
data = test_roi_positive[1:10000,], method = "class")
decisionTree
```

Рисунок 1.1. – Функція дерева

```
n= 10000

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 10000 1093 1 (0.89 0.063 0.024 0.0074 0.013 0.0022)
   2) campaign_id=W1,W11,W13,W14,W16,W18,W2,W20,W21,W23,W24,W25,W27,W28,W29,W3,W32,
W33,W34,W35,W36,W37,W38,W39,W40,W6,W7 9072  264 1 (0.97 0.029 0.00033 0 0 0)
     4) installs< 22.5 8147   73 1 (0.99 0.0086 0.00037 0 0 0) *
     5) installs>=22.5 925  191 1 (0.79 0.21 0 0 0 0)
      10) campaign_id=W1,W14,W18,W23,W25,W3,W34,W35,W36,W40,W6 763   59 1 (0.92 0.0
77 0 0 0 0) *
      11) campaign_id=W24,W27 162   30 3 (0.19 0.81 0 0 0 0)
        22) ttr< 0.07075 22    0 1 (1 0 0 0 0 0) *
        23) ttr>=0.07075 140    8 3 (0.057 0.94 0 0 0 0) *
   3) campaign_id=W10,W22,W31,W4,W8 928  563 3 (0.11 0.39 0.26 0.08 0.14 0.024)
     6) campaign_id=W10,W22,W31,W8 351  107 3 (0.24 0.7 0.063 0.0028 0 0)
      12) avg_cpt< 0.69915 138   75 1 (0.46 0.46 0.08 0.0072 0 0)
        24) avg_cpt>=0.32415 93    37 1 (0.6 0.3 0.086 0.011 0 0) *
        25) avg_cpt< 0.32415 45   10 3 (0.16 0.78 0.067 0 0 0) *
      13) avg_cpt>=0.69915 213   32 3 (0.099 0.85 0.052 0 0 0) *
     7) campaign_id=W4 577  357 5 (0.026 0.21 0.38 0.13 0.22 0.038)
      14) spend< 14.035 359  200 5 (0.042 0.33 0.44 0.18 0.011 0)
        28) impressions>=11.5 171   58 3 (0.064 0.66 0.19 0.064 0.023 0) *
        29) impressions< 11.5 188   61 5 (0.021 0.021 0.68 0.28 0 0) *
      15) spend>=14.035 218   96 9 (0 0.018 0.28 0.041 0.56 0.1)
        30) redownloads< 225.5 149   83 9 (0 0.027 0.41 0.054 0.44 0.067)
          60) avg_cpt< 0.7805 63   23 5 (0 0.063 0.63 0.095 0.21 0) *
          61) avg_cpt>=0.7805 86   33 9 (0 0 0.24 0.023 0.62 0.12) *
        31) redownloads>=225.5 69   13 9 (0 0 0 0.014 0.81 0.17) *
> |
```

Рисунок 1.2 – Дерево рішень у табличній формі

Коефіцієнти аргументів мають такий вигляд

```
> decisionTree$variable.importance
  campaign_id new_downloads      installs         spend   impressions
 1069.3501062   207.2113119   197.0231089   195.9165997   191.9305713
         taps   redownloads       avg_cpt           ttr       avg_cpa
  188.9838640   116.5659455    80.0925943    77.1771767    50.7773663
           cr
    0.2878839
> |
```

Рисунок 1.3 – Коефіцієнт аргументів

Візуально модель має вигляд:

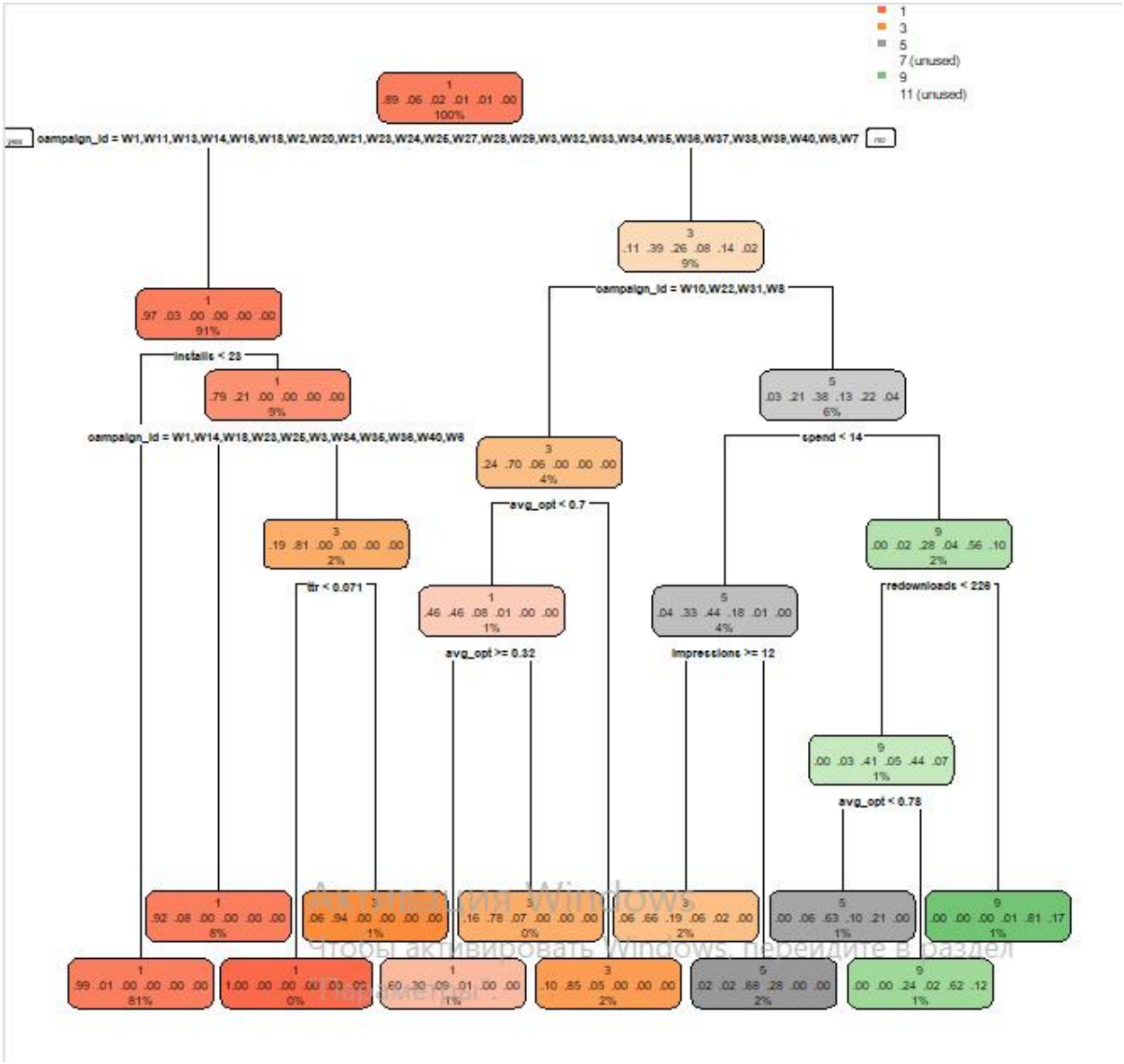

Рисунок 1.4 – Візуалізація дерева рішень за допомогою функцій **functionRpartPlot**

Далі модель була перевірена на точність на 3 типах даних, на навчальній вибірці, на тренувальній, та на тестовій вибірці. Навчальними даними стали перші 10000 даних. Тренувальна вибірка склала ще 3298 даних з вже наявної вибірки. Тестова вибірка складається з тих даних, у яких ROI був негативним.

Точність, яка була продемонстрована на навчальній вибірці складала 95,93%.

```
> paste('Accuracy =', accuracy_train)
[1] "Accuracy = 0.9593"
```

Рисунок 1.5 – Точність моделі дерева рішень на навчальній вибірці

Точність, яка була продемонстрована на тренувальній вибірці складала 95,42%.

```
> paste('Accuracy =', accuracy_train1)
[1] "Accuracy = 0.954228554107305"
```

Рисунок 1.6 – Точність моделі дерева рішень на тренувальні вибірці

Точність, яка була продемонстрована на тестовій вибірці складала 96,35%.

```
> paste('Accuracy =', accuracy_train2)
[1] "Accuracy = 0.963514162265963"
```

Рисунок 1.7 – Точність моделі дерева рішень на тестовій вибірці

Частина 2. Реалізація перехресної перевірки та випадкового лісу

Наступним метод - випадковий ліс. Код моделі випадкового лісу продемонстрований у додатку Б. Його модель реалізовувалися через функцію randomForest, де функція та аргументи були підібрані відповідно до тих, які були у випадкових дерев.

```
Call:
 randomForest(formula = as.factor(near_cpt_bid) ~ campaign_id +      spend +
 taps + installs + impressions + ttr + cr + avg_cpa +      avg_cpt + new_dow
nloads + redownloads, data = test_roi_positive[1:10000,      ], importance =
 TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 3.81%
Confusion matrix:
      1   3   5  7  9 11 class.error
1  8865  35   7  0  0  0 0.004715392
3   151 455  18  1  1  0 0.273162939
5     7  25 177 18 18  0 0.277551020
7     2   9  43 17  3  0 0.770270270
9     0   7  16  0 95  8 0.246031746
11    0   0   1  0 11 10 0.545454545
>
```

Рисунок 2.1 – Модель випадкового лісу

Далі за допомогою функції predict було знайдено точність та похибку моделі на різних вибірках.

Точність, яка була продемонстрована на навчальній вибірці складала 99,79%.

```
> paste('Accuracy =', round(conMatRF$overall["Accuracy"],5))
[1] "Accuracy = 0.9979"
```

Рисунок 2.2 – Точність моделі лісу на навчальній вибірці

Точність, яка була продемонстрована на тренувальній вибірці складала 90,78%.

```
> paste('Accuracy =', round(conMatRF1$overall["Accuracy"],5))
[1] "Accuracy = 0.90785"
```

Рисунок 2.3 – Точність моделі лісу на тренувальні вибірці

Точність, яка була продемонстрована на тестовій вибірці складала 91,83%.

```
> paste('Accuracy =', round(conMatRF2$overall["Accuracy"],5))
[1] "Accuracy = 0.91831"
```

Рисунок 2.4 – Точність моделі лісу на тестовій вибірці

Останнім був метод перехресної перевірки. Код моделі перехресної перевірки продемонстрований у додатку Б. Для нього треба розділити дата сет на **samp**, а саме на **trainData** та **validData**. За допомогою функції **trainControl** для якої потрібно вказати кількість повторень. Наступним кроком було написання функції **train**

За допомогою функції **format** було зроблено оцінку точності, яке склала 76,73%.

```
> paste("Estimated accuracy:", model_accuracy)
[1] "Estimated accuracy: 0.7673"
```

Рисунок 2.5 – точність моделі перехресної перевірки

Далі, була побудована таблиця з точністями моделей на різних вибірках.

Models	Predictors	Accuracy on Learn	Accuracy on Train	Accuracy on Test
Tree	data	0.96	0.95	0.96
RandomForest	data1	1.00	0.91	0.92
CrossValidation	data2	0.77	0.00	0.00

Рисунок 2.6 – точності всіх моделей на 3 видах вибірок

Також точності були створені у вигляді діаграми

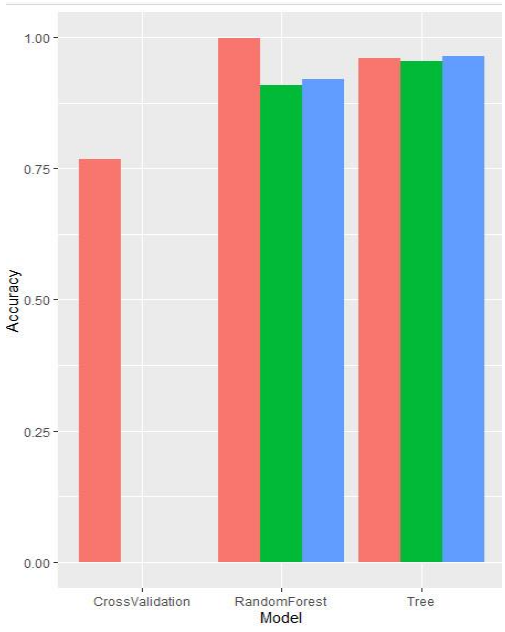

Рисунок 2.7– точності всіх моделей на 3 видах вибірок (графічно)

Останнім кроком, серед усіх моделей було обрано найкращу для цих даних, а саме, з відсутністю переобладнання, та з найвищою точністю. У даному випадку, цією моделлю виявилася модель випадкового дерева. І отже, прогнозовані дані cpt bid на вибірці з негативним ROI, були записані у новостворений стовпець з назвою *near_cpt_bid_roipos*, де є можливість порівняти значення біду з негативним ROI, та тими значеннями, які необхідно встановити, щоб він став позитивним.

Висновки

У роботі було проведено аналіз двох датасетів. В основу першого з них покладено дані про розміщення реклами додатку у Apple Add Search. Цей датасет включає 24 полів. Другий датасет, який у майбутній роботі для зручності об’єднаний з першим, включав аналіз ефективності реклами. Сама робота присвячена прогнозуванню того, яку бід ставку краще встановлювати при конкретних даних полів для підтримання ROI-позитивної моделі у Apple Ads Search.

На основі попередніх датасетів створено три моделі, а саме, модель дерева рішень, перехресна перевірка та випадковий ліс. В результаті навіть модель лісу, яка є найпростішою, серед вказаних трьох показала високу ефективність на навчальному наборі даних точність склала 95,93%, такий високий результат був забезпечений тим, що розмір самої вибірки є достатнім для аналізу ефективної bid-ставки. На тренувальному наборі даних була продемонстрована точність у 95,42%. А тестовий набір даних показав точність у 96,35%. Також подібні результати показали і дві інші перевірки, про які говорилось раніше. Точність моделі RandomForest склала 99,79% на навчальній вибірці, на тренувальній, її точність склала 90,78%, а на тестовій 91,83%. Щодо точності останньої моделі, а саме перехресної перевірки то вона склала 76,73%.

Тож зважаючи на результати, у даній моделі, на першому етапі, серед трьох моделей було викинуто модель перехресної перевірки, через недостатню точність на навчальній вибірці. Наступним кроком серед випадкового лісу та дерева для прогнозування значень в ROI негативній моделі було обрано дерево, оскільки різниця у 10% точності між результатами у навчальній та тренувальній вибірці, говорить про проблему переобладнання лісу. І вже за допомогою дерева був створений прогноз оптимальної бід ставки для створення ROI-позитивної моделі.

Для побудови моделі використано Rstudio. Для створення моделей були використано такі основні бібліотеки, як **rpart** та **caret**. Самі ж моделі створювалися такими функціями. Для моделі дерева використано функцію **rpart**. Для створення моделі випадкового дерева використовувався **randomForest**. А для створення перехресної перевірки – функція **train**.

Список використаної літератури

1. Мюллер, А. & Гвидо, С. (2016). Введение в машинное обучение с помощью Python. Москва.
2. Джеймс, Г & Уиттон, Д. & Хасти, Т.& Тибширани, Р. (2016). Введение в статистическое обучение с примерами на языке R. Москва: ДМК Пресс.
3. Geurts, P. Ernst, D. Wehenkel, L. (2006). Extremely randomized trees. Machine Learning (pp. 3-42)
4. Rubinfeld, D. Pindyck R. (2018). Microeconomics, 9th Edition. Upper Saddle River, N.J.: Pearson/Prentice Hall.
5. Gangmin, L. (2021). Do A Data Science Project in 10 Days.
6. Методи дерев рішень, класифікації та прогнозування [Електронний ресурс]. Режим доступу: https://moodle.znu.edu.ua/pluginfile.php?file=/486136/mod_resource/content/1/%d0%9b%d0%b5%d0%ba%d1%86%d1%96%d1%8f%209.pdf
7. Сравнение Apple Search Ads Advanced и Basic [Електронний ресурс]. Режим доступу: https://searchads.apple.com/ru/help/get-started/0001-compare-apple-search-ads-solutions
8. Пять причин попробовать Apple Search Ads для продвижения своего iOS-приложения [Електронний ресурс]. Режим доступу: https://kod.ru/apple-search-ads-dlya-prodvizhenia/
9. Випадкові дерева (Random Forest) [Електронний ресурс]. Режим доступу: https://dyakonov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/

10. Подробно про ROI и ROAS: как правильно анализировать рекламные кампании [Електронний ресурс]. Режим доступу: https://pengstud.com/blog/roi-i-roas/

Додаток А

(Обов'язковий)

Робота з даними

```
#libraries

library(rpart)

library(caret)

library(rpart.plot)

library(rattle)

library(expss)

library(randomForest)

library(tidyr)


#imports data from necessary data set

test <- read.csv("C:/work/dip/data/asa_dataset_final_11112021.csv", header =
TRUE, sep = ",")

roi <- read.csv("C:/work/dip/data/roi_by_country_11112021.csv", header = TRUE,
sep = ",")


#creating new data set for better manipulating with data

test1 <- data.table(test)

test2 <- data.table(test)

roi1 <- data.table(roi)
```

```
#creating new column, that turn cpt bid from continuous to discrete data
test1 <- test1[, near_cpt_bid:=0]
test1 <- test1[, rois:=0]
roi1 <- roi1[, roi_roi:=0]

#filter data set from useless data
test1$spend[test1$spend==0] <- NA
test1$installs[test1$installs==0] <- NA
test1$keyword_match_type[test1$keyword_match_type=="AUTO"] <- NA
test1<-test1[complete.cases(test1),]

#making cpt bid as a discret number
for(i in 1:dim(test2)[1]){
  if(test2$cpt_bid[i]>=0 && test2$cpt_bid[i] <= 2){
    test2$near_cpt_bid[i]=1
  }
  if(test2$cpt_bid[i]>2 && test2$cpt_bid[i] <= 4){
    test2$near_cpt_bid[i]=3
  }
  if(test2$cpt_bid[i]>4 && test2$cpt_bid[i] <= 6){
    test2$near_cpt_bid[i]=5
  }
```

```
  if(test2$cpt_bid[i]>6 && test2$cpt_bid[i] <= 8){
    test2$near_cpt_bid[i]=7
  }
  if(test2$cpt_bid[i]>8 && test2$cpt_bid[i] <= 10){
    test2$near_cpt_bid[i]=9
  }
  if(test2$cpt_bid[i]>10 && test2$cpt_bid[i] <= 12){
    test2$near_cpt_bid[i]=11
  }
  if(test2$cpt_bid[i]>12 && test2$cpt_bid[i] <= 14){
    test2$near_cpt_bid[i]=13
  }
}

#fill roi data set new column with 0 or 1 depends on roi_6m_percentage column
for(i in 1:dim(roi)[1]){
  if(roi$roi_6m_percentage[i]>=0){
    roi1$roi_roi[i]=1
  }
  if(roi$roi_6m_percentage[i]<0){
    roi1$roi_roi[i]=0
  }
```

```
}

#transfer roi_roi in roi dataset to test dataset
for(i in 1:dim(test1)[1]){
  for(j in 1:dim(roi1)[1]){
  if(test1$date[i]==roi1$date[j] &&
test1$country_or_region[i]==roi1$Country_code[j]){
    test1$rois[i]=roi1$roi_roi[j]
    }
  }
}

#creating data set with positive ROI
test_roi_positive <- data.table(test1)
test_roi_positive$rois[test_roi_positive$rois==0] <- NA
test_roi_positive<-test_roi_positive[complete.cases(test_roi_positive),]

#creating data set with negative ROI
test_roi_negative <- data.table(test1)
test_roi_negative$installs[test_roi_negative$rois!=0] <- NA
test_roi_negative<-test_roi_negative[complete.cases(test_roi_negative),]
```

Додаток Б

(Обов’язковий)

Створення моделей

```
#creating the model of decision tree

set.seed(1234)

decisionTree <- rpart(near_cpt_bid ~ campaign_id+spend+taps+installs+

                      impressions+ttr+cr+avg_cpa+avg_cpt+

                      new_downloads+redownloads,

data = test_roi_positive[1:10000,], method = "class")

decisionTree

decisionTree$variable.importance


#calculating the accuracy on learn data

Accuracy_train <- predict(decisionTree, test_roi_positive[1:10000,], type =
"class")

conMat <- confusionMatrix(as.factor(Accuracy_train),
as.factor(test_roi_positive[1:10000,]$near_cpt_bid))

accuracy_train <- conMat$overall["Accuracy"]

paste('Accuracy =', accuracy_train)


#calculating the accuracy on train data

Accuracy_train1 <- predict(decisionTree, test_roi_positive[10000:13298,], type =
"class")
```

```
conMat1 <- confusionMatrix(as.factor(Accuracy_train1),
as.factor(test_roi_positive[10000:13298,]$near_cpt_bid))

accuracy_train1 <- conMat1$overall["Accuracy"]

paste('Accuracy =', accuracy_train1)


#calculating the accuracy on testing data

Accuracy_train2 <- predict(decisionTree, test_roi_negative, type = "class")

conMat2 <- confusionMatrix(as.factor(Accuracy_train2),
as.factor(test_roi_negative$near_cpt_bid))

accuracy_train2 <- conMat2$overall["Accuracy"]

paste('Accuracy =', accuracy_train2)


#drawing different graph which describe tree

fancyRpartPlot(decisionTree, caption = "")

rpart.plot(decisionTree)


#creating the model of random forest

test1$near_cpt_bid <- as.numeric(test1$near_cpt_bid)

RF_model1 <- randomForest(as.factor(near_cpt_bid) ~
campaign_id+spend+taps+installs+impressions+ttr+cr+avg_cpa+avg_cpt+new_do
wnloads+redownloads, data=test_roi_positive[1:10000,], importance=TRUE)

RF_model1
```

```
#calculating the accuracy on learn data

RF_prediction1 <- predict(RF_model1, test_roi_positive[1:10000,])

conMatRF<- confusionMatrix(RF_prediction1,
factor(test_roi_positive[1:10000,]$near_cpt_bid))

paste('Accuracy =', round(conMatRF$overall["Accuracy"],5))


#calculating the accuracy on train data

RF_prediction2 <- predict(RF_model1, test_roi_positive[10000:13298,])

conMatRF1 <- confusionMatrix(RF_prediction2,
factor(test_roi_positive[10000:13298,]$near_cpt_bid))

paste('Accuracy =', round(conMatRF1$overall["Accuracy"],5))


#calculating the accuracy on testing data

RF_prediction3 <- predict(RF_model1, test_roi_negative)

conMatRF2 <- confusionMatrix(RF_prediction3,
factor(test_roi_negative$near_cpt_bid))

paste('Accuracy =', round(conMatRF2$overall["Accuracy"],5))


#dividing set to train and test data

set.seed(1000)

samp <- sample(nrow(test_roi_positive), 0.8 * nrow(test_roi_positive))

trainData <- test_roi_positive[samp, ]
```

```
validData <- test_roi_positive[-samp, ]

#choose how many tests and dividing it will be
set.seed(3214)
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 5,
                        search = "grid")

#creating model where tree will use cross validation to improve result
crossValidation <- train(near_cpt_bid ~
campaign_id+spend+taps+installs+impressions+ttr+
                           cr+avg_cpa+avg_cpt+new_downloads+redownloads
                         , data = test_roi_positive,
                                  method = "rpart",
                                  trControl = control)
crossValidation

#calculate model accuracy
model_accuracy <- format(crossValidation$results$Rsquared[1], digits = 4)
paste("Estimated accuracy:", model_accuracy)
```

```
#creating tabble with all accuracies

Model <- c("Tree","RandomForest","CrossValidation")

Learn <- c(0.9593, 0.9979, 0.7673)

Train <- c(0.9542, 0.9081, 0)

Test <- c(0.9635, 0.9191, 0)

Pre <- c("data", "data1", "data2")

df1 <- data.frame(Model, Pre, Learn, Train, Test)

df2 <- data.frame(Model, Learn, Train, Test)

knitr::kable(df1, longtable = TRUE, booktabs = TRUE, digits = 2, col.names
=c("Models", "Predictors", "Accuracy on Learn", "Accuracy on Train", "Accuracy
on Test"),

          caption = 'The Comparision among 3 models'

)


#creating bar chart with all accuracies

df.long <- gather(df2, Dataset, Accuracy, -Model, factor_key =TRUE)

ggplot(data = df.long, aes(x = Model, y = Accuracy, fill = Dataset)) +

geom_col(position = position_dodge())


#writing down cpt bids, which are better fitting to make roi negative data to
positive

test_roi_negative <- test_roi_negative[, near_cpt_bid_roipos:=0]
```

```
test_roi_negative$near_cpt_bid_roipos<-Accuracy_train2
```