НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Система інтелектуального аналізу даних для прогнозування відтоку аудиторії в телекомунікаційній галузі»

Виконала:
студентка IV курсу, групи КА-92
Гаврилко Дар'я Олегівна _____

Керівник:
доц., к.т.н. Дідковська М. В. _____

Консультант з економічного розділу:
доц., к.е.н. Рощина Н.В. _____

Консультант з нормоконтролю:
доц.к.т.н. Статкевич В.М. _____

Рецензент:
доц., к.т.н. Заболотня Т. М. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка _____

Київ 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

НН Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2023 р.

ЗАВДАННЯ

на дипломну роботу студентці

Гаврилко Дар'ї Олегівні

1. Тема роботи «Система інтелектуального аналізу даних для прогнозування відтоку аудиторії в телекомунікаційній галузі», керівник роботи доцент, к.т.н. Дідковська М.В., затверджені наказом по університету від «___»_____ 2023 р. № 2065-с

2. Термін подання студентом роботи

3. Вихідні дані до роботи: статистичні дані про користувачів телекомуніка-

ційної компанії.

4. Зміст роботи: аналіз предметної області, дослідження даних, вибір моделей машинного навчання для подальшої роботи, побудова обраних моделей та створення прогнозу, аналіз отриманих результатів та вибір найкращої моделі.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доцент, канд. екон. наук Рощина Н.В.		

7. Дата видачі завдання: _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми ДР	17.04.2023-23.04.2023	виконано
2	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	17.04.2023-23.04.2023	виконано
3	Ознайомлення з ДСТУ 3008-95 та стандарти ЄСПД	17.04.2023-28.04.2023	виконано
4	Проведення дослідження за темою БДР під керівництвом керівника	17.04.2023-28.04.2023	виконано
5	Завершення роботи над першим варіантом частини БДР	28.04.2023-16.05.2023	виконано
6	Проведення роботи над теоретичною частиною БДР	17.05.2023-29.05.2023	виконано
7	Проведення роботи над програмним продуктом	30.05.2023-14.05.2023	виконано
8	Оформлення БДР та аналіз отриманих результатів	15.05.2023-21.05.2023	виконано

Студентка _____ Дар’я ГАВРИЛКО

Керівник _____ Марина ДІДКОВСЬКА

РЕФЕРАТ

Дипломна робота: 147 с., 69 рис., 7 табл., 2 додатки, 37 джерел.

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, МАШИННЕ НАВЧАННЯ, БІНАРНА КЛАСИФІКАЦІЯ, МОДЕЛЮВАННЯ, ПРОГНОЗУВАННЯ ВІДТОКУ.

Об’єкт дослідження – статистичні дані про користувачів телекомунікаційної компанії.

Предмет дослідження – методи обробки статистичних даних та моделі машинного навчання, зокрема, логістична регресія, наївний баєсівський класифікатор, дерева рішень, метод опорних векторів та XGBoost.

Мета дипломної роботи – побудова системи прогнозування відтоку аудиторії в телекомунікаційній галузі на основі методів машинного навчання.

Актуальність – розробка ефективних інструментів прогнозування та управління відтоком користувачів, з метою збереження аудиторії, покращення їх користувацького досвіду та корегування стратегії розвитку телекомунікаційної компанії в цілому.

В даній дипломній роботі було проведено створено програмний продукт мовою Python. А саме, було виконано побудову прогностичні моделі, проведено аналіз отриманих результатів та вибір найкращої моделі. Особливу увагу було приділено підготовці даних та їх попередньому аналізу, з метою визначення патернів поведінки користувачів та виявлення важливих факторів, що потенційно впливають на відтік.

В подальшому можна підвищувати прогностичну здатність моделей та впроваджувати їх в реальну систему, що може включати інтеграцію в існуючі системи управління клієнтами, автоматизовані системи маркетингу та веб-платформи для моніторингу відтоку.

ABSTRACT

Bachelor’s thesis: 147 p., 69 fig., 7 tabl., 2 appendices, 37 sources.

INTELLECTUAL DATA ANALYSIS, MACHINE LEARNING, BINARY CLASSIFICATION, MODELING, CHURN PREDICTION.

Object of research - telecommunication company user’s statistical data.

Subject of research - methods of statistical data processing and machine learning models, including logistic regression, naïve Bayes classifier, decision trees, support vector machine and XGBoost.

The aim of the thesis is to build a churn prediction system in the telecommunications industry based on machine learning methods.

Relevance - development of effective prediction and churn management tools to retain customers, improvement of their experience and adjustment of the overall development strategy of the telecommunications company.

In this thesis, a software product was developed using the Python programming language. In particular, predictive models were designated and the obtained results were analyzed to select the best model. Particular attention was paid to data preparation and preliminary analysis to identify user behavior patterns and important factors that may potentially influence the churn.

In the future, the predictive capability of the models can be improved and they can be implemented into an existing system, which may involve integration into existing customer management systems, automated marketing systems and web platforms for churn monitoring.

ЗМІСТ

ВСТУП

Телекомунікаційна галузь стала невід'ємною складовою сучасної економіки та відіграє ключову роль у розвитку суспільства та бізнесу. Ринок телекомунікацій постійно зростає та розширюється. Швидкий розвиток технологій, інноваційні рішення та зростання популярності цифрових сервісів призводять до стрімкого збільшення попиту на послуги зв'язку та передачі даних. Саме телекомунікаційна галузь стала каталізатором розвитку сучасного цифрового суспільства, яке є залежним від стабільного з'єднання та швидкого доступу до інформації. Конкуренція в телекомунікаційній галузі є особливо жорсткою, велика кількість компаній кожного дня змагаються за лояльність споживачів, які, в свою чергу, стають все більш вимогливими до якості отримуваних послуг і готові змінювати постачальника, якщо ті не задовольняють їх потреби.

Тож, швидкий технологічний прогрес і конкурентна боротьба змушують компанії постійно удосконалюватись та розширювати свої можливості, а діяти лише на основі інтуїції та досвіді вже не є достатньо ефективним. Сьогодні підприємства націлені на розширення своєї клієнтської бази, але що більш важливо, збереження вже існуючої аудиторії, яка є одним з ключових джерел прибутку. Втрата навіть частини користувачів може призвести до значних фінансових збитків і зниження позицій на ринку. Саме тому компаніям критично важливо мати ефективні інструменти для керування відтоком аудиторії. Розробка стратегій збереження і залучення нових клієнтів, а також виявлення факторів, що спонукають їх до зміни постачальника послуг вимагає від компаній постійно аналізувати великі об’єми даних.

Ефективним підходом до вирішення поставленої задачі є застосування машинного навчання, яке дає можливість аналізувати великі обсяги даних, визначати залежності та патерни в поведінці користувачів. Моделі машинного навчання здатні автоматично виявляти складні шаблони та тренди, що робить

їх незамінними інструментами для розуміння та передбачення відтоку. Отримані прогнози дозволяють компаніям приймати обґрунтовані рішення та впроваджувати ефективні стратегії з метою збереження клієнтів та розвитку бізнесу в цілому.

Тож, у даній роботі буде розглянута задача прогнозування відтоку користувачів як задача бінарної класифікації. З цією метою буде використано різні моделі машинного навчання і виконано порівняльний аналіз цих моделей для визначення тої, яка забезпечити максимальну точність прогнозування.

.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Телекомунікації, як ключовий фактор економічної трансформації та розвитку суспільства

Науково-технічна революція XX століття спричинила суттєві зміни у світовому економічному розвитку та викликала революційні трансформації в технологічному базисі виробництва. Цей процес сприяв становленню і розвитку інформаційного суспільства, та як наслідок інформаційно-комунікаційні технології набули визначальної ролі, з кожним роком все більше визначаючи обличчя нашої епохи. В їх складі важливу роль відіграють саме телекомунікації, що почали швидко розвиватися наприкінці XIX століття, згодом виникли такі поняття як телефонний обмін, несучі системи, ретранслятори, галузь повністю комп'ютеризували. Всі ці модифікації сприяли поширенню телекомунікаційних систем країнами світу, а обсяги передаваної інформації зростають з кожним роком, що в свою чергу підсилює процеси глобалізації в цілому. Інформаційні ресурси організацій, регіонів, країн стали так само стратегічно важливими, що й запаси енергії, сировини, копалин тощо.

Технології та послуги, що надаються цим сектором, стали повсякденною необхідністю для мільярдів людей по всьому світу, кожен другий житель планети має стільниковий телефон, а кожен п'ятий - вихід в Інтернет. А регулювання та реформування галузі стали предметом широких економічних дискусій. Рішення, що ухвалюються урядами щодо телекомунікацій, мають прямий вплив на соціальний та економічний добробут громадян. Телекомунікації проникають у всі процеси економічного виробництва і стають невід’ємною частиною будь-якої бізнес-діяльності, яка спрямована на продукування послуг і товарів для споживачів. На сьогодні інформаційно-

комунікаційні технології є новітнім та потужним ресурсом економічного зростання, що підтримує функціонування практично усіх галузей.

Завдяки новим технологіям передачі даних, таких як пакетна комутація, використання різних середовищ забезпечення і передавання мобільності зв’язку, нині є можливість суттєво підвищити ефективність, продуктивність і якість обслуговування в сфері. В умовах лібералізації ринків оператори зосередилися на пошуку нових ринкових форм комплексних рішень з метою розширення діапазону послуг, що пропонуються користувачам, а комбінація мобільного та фіксованого доступу, широкосмугового інтернету та передачі даних надають операторам зв'язку ідеальні умови для конкуренції та розвитку на ринку телекомунікацій.

1.2 Галузь телекомунікацій в Україні

Телекомунікації є сучасним, різноманітним сектором економіки, що стрімко розвивається у будь-якій країні, і Україна не є виключенням. Вони є невід’ємною частиною соціальної та виробничої інфраструктури України і призначена для задоволення потреб фізичних та юридичних осіб, органів державної влади в телекомунікаційних послугах[1].

Галузь не ізольована від інших. Скоріше її розвиток впливає на інші сфери соціально-економічного життя, часто визначаючи розвиток фінансового сектору економіки, електронної комерції, бізнес-процесів міжнародних відносин та оперативних комунікацій у глобальному економічному середовищі. Сьогодні телекомунікації стали одним із ключових секторів економіки України, більше 20% загального обсягу послуг, що надаються в країні, реалізується через діяльність операторів і провайдерів[2]. Ці дані свідчать про те, що сфера є однією із системоутворюючих в економіці, а ринок телекомунікацій України став потужним інструментом розвитку суспільства.

Телекомунікації в Україні стрімко розвивалися у пореформений період 1990-2010 рр. Попри економічний спад економіки в цих роках, галузь зв'язку, в цілому, розвивалася безкризово та демонструвала високі темпи зростання, що випереджали темпи зростання економіки в країні, частка галузі в структурі ВВП постійно збільшувалася, у 2005-2007 роках вона збільшилася з 3.1% до 6.5%. Особливо швидко розвиваються послуги по окремим сегментам ринку, а саме мобільний зв’язок та Інтернет – послуги, що тільки підтверджує неухильно зростаючу роль цього сектора в економіці України. А станом на 2008 рік рівень проникнення мобільного зв’язку в Україні досяг 120,1% , для порівняння в той же час у США ця позначка сягала 72%, темп зростання абонентської бази в Україні склав 117 % у 2009 р., а темп зростання вкладання всіх прямих іноземних інвестицій (ПІІ) економіки країни в тому ж році склав 7%. Вже у 2009 році оператори усіх форм власності надали послуг зв'язку на 46,1 млрд грн[3].

Реалізація послуг зв'язку в Україні розвивалася і у 2011 склала 50,3 млрд грн. Обсяги доходів від надання послуг зв'язку населенню склали майже 37% від загального обсягу послуг зв'язку[4].

У 2021 році Національна комісія з регулювання у сферах електронних комунікацій (далі НКРЗІ) оприлюднила звіти [5] за поточний рік. З яких можна зробити висновки, що український ринок телекомунікацій «стрибнув» досить високо. Діяльність НКРЗІ здійснювалась з метою сприяння ефективному функціонуванню конкурентного та відкритого ринку телекомунікацій, про що насамперед свідчить підвищення якості наданих телекомунікаційних послуг та збільшення їх обсягів. Приріст у доходах мобільного зв'язку та фіксованого доступу до інтернету становив 14%, а інтернет проникнення українських домогосподарств сягнуло позначки 90%. Драйвером такого росту прогнозовано стали послуги доступу до Інтернет мережі, була проведена масштабна розбудова мереж 4G у діапазоні 900 МГц. Завдяки ініціативі Міністерства цифрової трансформації: у межах активної субвенції для

віддалених населених пунктів сільської місцевості було побудовано приблизно 20 000 км оптоволоконних мереж по всій країні.

Трохи статистики за 2021 рік, доходи від надання послуг зв’язку склали 87 742 млн грн, при цьому, в структурі доходів найбільшу частку склали телекомунікаційні послуги, а саме 92,3 %. Відповідну статистику зображено на рисунку 1.1.

Рисунок 1.1 – Структура доходів від надання послуг зв’язку за 2021 рік., млн грн

За 2019-2021 роки доходи від надання телекомунікаційних послуг мали тенденцію до зростання, що зображено на рисунку 1.2. Зокрема, в 2021 році доходи збільшилися на 10% порівняно з 2020 роком.

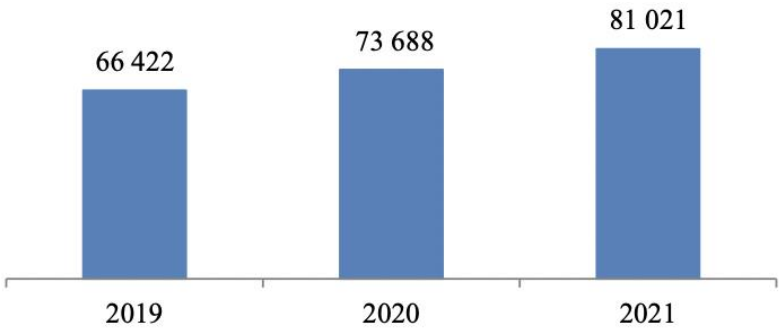


Рисунок 1.2 – Динаміка доходів від надання телекомунікаційних послуг за 2019-2021 роки, млн грн

У 2021 році, найбільш значущими складовими доходів від телекомунікаційних послуг були фіксований доступ до мережі Інтернет -

19,5% та рухомий (мобільний) зв'язок - 66,0%, це можна побачити на рисунку 1.3. Рухомий зв'язок має найвищу динаміку зростання, його частка у доходах збільшилась на 2,3% порівняно з 2020 роком.

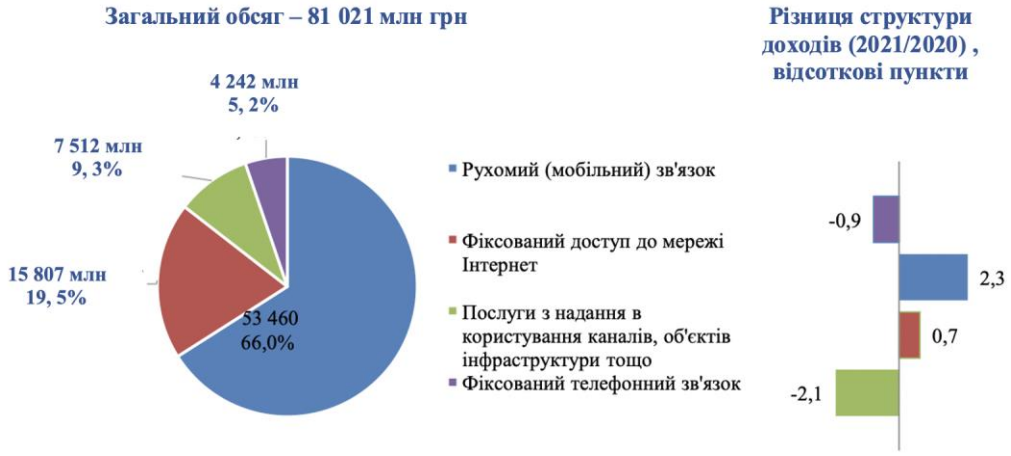


Рисунок 1.3 – Структура доходів від надання телекомунікаційних послуг за 2021 рік, млн грн

За останні чотири роки інвестиції у галузь збільшилися на 15% і склали 13,2 млрд грн. Це свідчить про успішну роботу індустрії та доводить, що вона заслуговує на підтримку та увагу з боку регулюючих та законодавчих органів.

Нові умови господарювання, пов'язані з пандемією та її наслідками, стимулювали зростання високотехнологічних та високорівневих виробництв. Телекомунікаційна галузь зараз має свої особливості, зумовлені багатьма факторами, наприклад цільовим ринком, що стрімко розвивається, активною інноваційною діяльністю, впливом конкурентів, умовами споживання продукту та його унікальністю. Сьогодні ринок телекомунікацій став на передову технологічного та цифрового розвитку України. Планується здійснення заходів для забезпечення подальшого зростання на основі мереж наступного покоління, що передбачає конвергенцію телекомунікаційних та інформаційних мереж та послуг. В рамках розвитку своєї діяльності, оператори мобільного зв’язку поступово впроваджують технологічні інновації, що розширює перелік послуг, які надаються споживачам. Вони здійснюють впровадження приватних мереж LTE, розширюються мережі IoT,

запроваджуються тестові зони 5G, все це сприяє збільшенню кількості споживачів даних послуг. На операторів зараз покладаються великі надії щодо забезпечення країни повноцінним мобільним та інтернет-зв'язком. Цього вимагає державна стратегія цифровізації, яка є частиною Національної економічної стратегії-2030.

У 2022 галузь зіткнулась з великою кількістю труднощів. За підрахунками Міжнародної телекомунікаційної спілки (ITU), що належить ООН, від початку повномасштабного вторгнення телекомунікаційна інфраструктура України зазнала збитків на $1,8 млрд. У звіті охоплені перші шість місяців війни та зазначається про значні руйнування і пошкодження телекомунікаційної інфраструктури у понад 10 з 24 регіонів України. Але попри все рівень інтернет-проникнення вдалося зберегти на рівні 2021-го, хоча ми досі можемо спостерігати проблеми з Інтернетом через обстріли та руйнування, середня швидкість інтернету по Україні продовжує зростати, і у 2022-му вона збільшилась 8,3 Мб/с у порівнянні попереднім роком.

Станом на початок 2023 року ситуація стабілізувалася, а український телеком став більш підготовленим до екстремального зростання чи перерозподілу трафіку. Тенденції ринку на 2023 рік визначають споживач і зовнішнє середовище, тож зв'язківці планують приділити багато уваги відмовостійкості мережі та збільшенню часу автономної роботи у разі аварійного вимкнення світла. А головна мета операторів не змінилась - забезпечити стійкий та надійний зв'язок в Україні, незважаючи на усі труднощі.

1.3 Проблема відтоку користувачів та його причини

Конкурентний ринок сьогодні диктує необхідність не тільки залучення нових клієнтів, але й збереження тих, що вже є. Змагаючись за увагу та лояльність користувачів, відтік аудиторії стає все більш актуальною проблемою для компаній у будь-якій галузі. Кожна втрачена людина - це не тільки втрачений дохід, але й негативний вплив на репутацію компанії. Тому питання прогнозування та запобігання відтоку клієнтів стає надзвичайно важливим для успішного функціонування бізнесу.

Відтік клієнтів (customer churn) - це процес втрати споживачів товарів та/або послуг протягом певного періоду часу. Високий відтік вказує на те, що значна частина користувачів припиняє співпрацю з підприємством. Коефіцієнт відтоку - це математичний розрахунок відсотка клієнтів, які навряд чи вчинять ще одну купівлю в компанії. Оцінка рівня відтоку та аналіз причин особливо важливим є для компаній, що працюють за передплатньою або транзакційною моделлю бізнесу, тобто компанії, що очікують регулярних платежів від клієнтів.

Тож, чому відтік має таке велике значення для бізнесу? За останнє десятиліття бізнес-середовище зазнало експоненційного зростання, і цей тренд не сповільнюється. Тепер компанії конкурують не з кількома, а з сотнями інших підприємств. Нові конкуренти з'являються з все привабливішими пропозиціями та випробовують лояльність клієнтів. Сучасний ринок переповнений продуктами та послугами, а маркетингові стратегії, напрямлені на їх просування, стрімко розвиваються та стають все більш жорсткими і ефективними. Привернути увагу клієнтів, витримати конкуренцію та забезпечити розвиток бізнесу стає надзвичайно складним завданням.

Компанії втрачають близько 1,6 трильйона доларів на рік через відтік клієнтів. Дослідження [6] за 2021 рік від KPMG, міжнародної мережі незалежних фірм, що надають аудиторські, податкові та консультаційні

послуги, виявило, що утримання клієнтів стало основним драйвером доходу компанії. Стверджується, що відбулася переломна зміна в способах конкуренції між компаніями. Успіх бізнесу вже не залежить від того, наскільки хороші продукти, чи якісний досвід користування пропонує компанія. Ключовим фактором успіху стає ефективна взаємодія з клієнтом протягом усього його життєвого циклу, що забезпечує постійну присутність компанії у житті споживачів, а новітні технології мають допомагають їм в цьому. Фактори, що впливають на прибутковість бізнесу, зображено на рисунку 1.4.

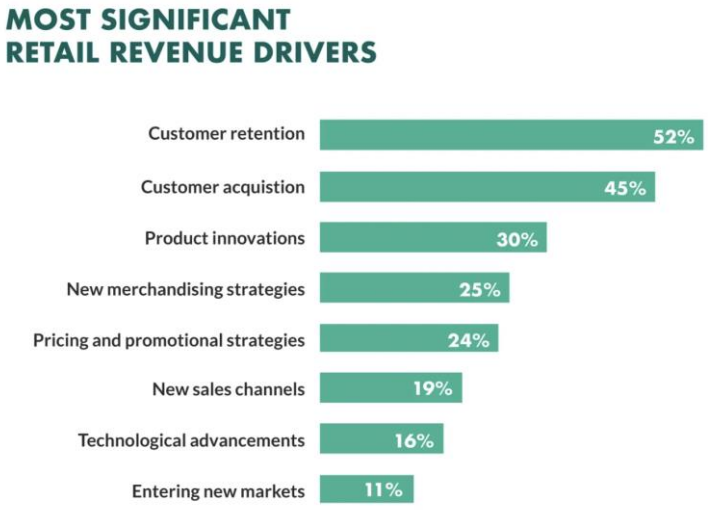


Рисунок 1.4 – Фактори, що впливають на прибутковість бізнесу

Існує фраза "знання - це сила, а дані - гроші" – глибоке розуміння потреб та поведінки клієнтів, яке базується на безперервному аналізі та обробці даних, робить її особливо актуальною. Компанії, зазначені в згаданому вище дослідженні, активно практикують подібний підхід, безперервно аналізуючи потік даних в режимі реального часу для ефективного взаємодії зі своєю цільовою аудиторією. Результатом є високий рівень задоволеності клієнтів, провідні позиції на ринку та процвітання бізнесу.

Переглянемо трохи статистики, за даними Forrester [7], залучення нових клієнтів коштує в 5 разів дорожче, ніж утримання існуючих. Щоб вивести нового клієнта на той самий рівень, що й існуючого, бізнесу знадобиться в 16

разів більше коштів. Окрім того, чим більше клієнтів утримує бізнес, тим більший дохід він отримує. Наприклад, у звіті Гарвардської бізнес-школи [8] стверджується, що навіть невелике підвищення рівня утримання клієнтів на 5% може призвести до значного збільшення прибутку компанії від 25% до 95%. І левова частка, а саме 65% бізнесу компанії, припадає на існуючих клієнтів.

Але і це ще не все, за даними Gartner [9], 80% майбутніх прибутків компанії залежать лише від 20% її існуючих клієнтів. Це ще раз підкреслює важливість утримання і задоволення існуючих клієнтів для успіху бізнесу. Тим часом Marketing Metrics [10] стверджує, що ймовірність продажу продукту чи послуги існуючому клієнту становить 60-70%, і лише 5-20% - новому потенційному клієнту.

Причини, які спонукають користувачів припинити співпрацю з компанією - різні, проте користувачів умовно можна віднести до одного з трьох типів відтоку.

1. Природний – це той випадок, коли у користувача змінились потреби або життєві обставини, що може бути пов’язано з переїздом, зміною місця проживання, хворобою, смертю чи зміною способу життя.

2. Усвідомлений – у його основі лежить свідомий вибір на користь іншого продукту, який його більше влаштовує за вартістю, якістю, рівнем сервісу тощо. Клієнт вибирає зручніший та вигідніший варіант та йде до конкурента.

3. Прихований – не зовсім відтік, швидше зниження активності, тимчасове навідування до конкурентів, пошук кращих. Компанія продовжує забезпечувати послугами своїх клієнтів, проте останні користуються ними рідше або в менших обсягах. Цей сегмент аудиторії не враховується при розрахунку рівня відтоку, оскільки вони не вважаються повністю втраченими. Проте це важлива частина клієнтської бази, оскільки відмова від послуг зазвичай відбувається поступово, а отже існує можливість активізувати цю частину аудиторії та зберегти їх.

Як вже було зазначено вище, причини відтоку клієнтів можуть бути найрізноманітнішими. На рисунку 1.5 продемонстровано основні причини відтоку клієнтів, згідно досліджень Forum Corporation [11].

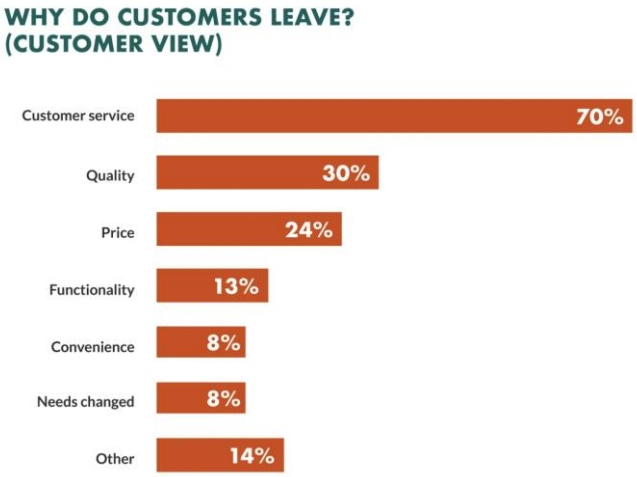


Рисунок 1.5 – Причини відтоку клієнтів

Тож факторами, які впливають на відтік, може бути: якість продуктів, користувач закриває очі на багато речей заради відповідної якості сервісу за який він платить; негативний досвід обслуговування; цікавість, яка може бути зумовлена модними тенденціями чи рекомендаціями знайомих та спонукати користувача спробувати щось нове; обставини, наприклад, зміна фінансового становища; поява продуктів замінників, які дозволяють задовольнити потреби клієнтів. Характерним прикладом є тенденція останніх років до зменшення кількості абонентів фіксованого телефонного зв'язку. Як наслідок, станом 2021 рік кількість ліній фіксованого телефонного зв'язку знизилась на 31,1 % порівняно з попереднім роком. Статистика зображена на рисунку 1.6.

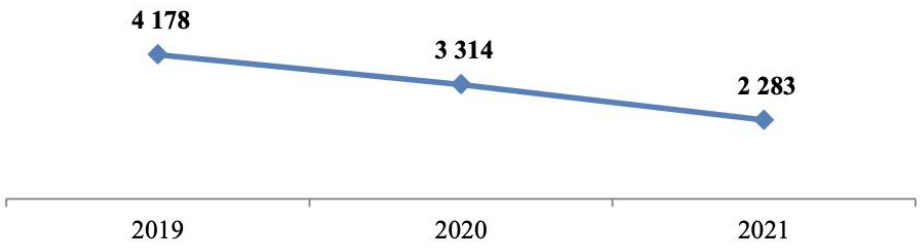


Рисунок 1.6 – Кількість абонентів фіксованого телефонного зв'язку, тис. од.

Широке поширення комп'ютерних програм та мобільних додатків стало основним фактором, що призвів до такої тенденції. Це дозволяє споживачам послуг доступу до Інтернету замінити традиційний фіксований телефонний зв’язок мобільним або такими сервісами як Skype, Google Meet, Zoom, Microsoft Teams, Whats App, Telegram та інші.

1.4 Проблема відтоку користувачів у галузі телекомунікацій

Актуальність проблеми відтоку для бізнесу вже обгрунтована вище, однак не можна не зупинитись детальніше на цій проблемі в сфері телекомунікацій. Середній показник утримання клієнтів у телекомунікаційній галузі становить 69%. Хоча це не найгірший показник, рівень відтоку клієнтів у телекомунікаційному секторі є одним з найвищих порівняно з іншими галузями. Прибутковість бізнесу значною мірою залежить від лояльності клієнтів, яку в цій індустрії буває важко отримати. Щорічна вартість відтоку клієнтів становить 4 мільярди доларів в Європі та Америці, а в усьому світі - 10 мільярдів доларів. [12] За останні кілька десятиліть телекомунікаційна галузь стала свідком величезного зростання і розвитку з точки зору технологій, рівня конкуренції, кількості операторів, нових продуктів, послуг тощо, а переключитися на нового провайдера можна за лічені хвилини. Тож, попит на послуги телекомунікаційної галузі високий, але і пропозиція теж. Станом на 2022 рік у США налічуватиметься приблизно 747 телекомунікаційних компаній. Це означає, що якщо клієнти захочуть змінити компанію, у них буде більш ніж достатньо альтернатив для вибору. В той час в Україні за офіційними даними НКРЗІ [13] на початок 2022 року діє більше 200 телекомунікаційних компаній, які надають різноманітні послуги. Серед них є національні оператори так званої «великої трійки» - Київстар, Vodafone, lifecell, та регіональні оператори, які пропонують послуги у певних областях

країни. До прикладу, розглянемо українського телекомунікаційного оператора lifecell, що надає широкий спектр послуг мобільного зв'язку та передачі даних. Компанія активно розвиває інноваційну екосистему у галузі телекомунікацій, реалізуючи проєкти, спрямовані на створення «розумного міста», розбудови мереж IoT та покращення безпеки громадян. Та все ж станом на кінець 2022 року активна 3-місячна абонентська база lifecell зменшилась на 7.6% та склала 8.5 млн користувачів[14]. Операційні дані компанії lifecell можна побачити на рисунку 1.7.

ОПЕРАЦІЙНІ ПОКАЗНИКИ

Операційні дані lifecell	Квартал			Рік		
	4 кв. 21	4 кв. 22	Річний % змін	2021	2022	Річний % змін
Кількість абонентів, млн.[1]	**10.1**	**10.2**	***1.0%***	**10.1**	**10.2**	***1.0%***
Активних (3 місяці)[2]	9.2	8.5	*(7.6%)*	9.2	8.5	*(7.6%)*
MoU хвилини (12 місяців)	**179.0**	**148.0**	***(17.3%)***	**180.9**	**156.9**	***(13.3%)***
ARPU (щомісячний дохід на абонента) змішаний, грн.	**80.2**	**86.0**	***7.2%***	**73.7**	**77.1**	***4.6%***
Активних (3 місяці), грн.	88.5	104.5	*18.1%*	83.2	91.5	*10.0%*

Рисунок 1.7 – Операційні дані lifecell

Конкуренція зараз на рекордно високому рівні, компанії критично важливо відстежувати відтік клієнтів. Адже розуміння поведінки свого користувача, рівня відтоку, основних факторів, які на це впливають, допомагає їй корегувати стратегію розвитку та утримувати позиції на ринку. На основі результатів проведеного аналізу даних діджитал-оператор lifecell пропонує нові вигідні та зручні тарифи/сервіси/послуги для користувачів розумних пристроїв. До цифрового портфоліо компанії, яке вже покращує користувацький досвід, входять: багатофункціональний месенджер BiP, хмарне сховище lifebox, мобільний платіжний сервіс, застосунки TV+, lifecell music тощо.

Причини відтоку вже були вказані, але для телекомунікаційної галузі найпоширенішими є наступні.

1. Погана якість продуктів та послуг. Перша і найбільш очевидна причина втрати аудиторії, яка спонукає її покинути компанію і перейти до

іншого оператора, - це те, що сервіс, не відповідає їхнім очікуванням. Наприклад, якщо мережа страждає від неякісного обслуговування, великих простоїв або ненадійності, цілком логічно, що клієнти можуть шукати іншого постачальника послуг.

2. Неякісне обслуговування. Негативний досвіт комунікації з представниками компанії є наступною важливою причиною, чому вони можуть розірвати контракт та почати пошуки нової компанії. Клієнти хочуть, щоб обслуговування було доступним, ефективним і персоналізованим.

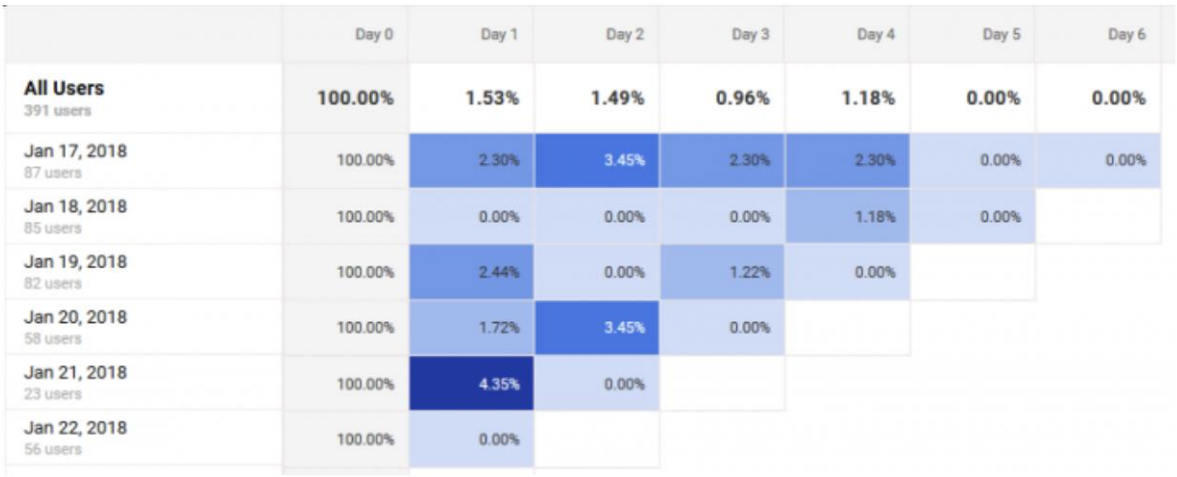
3. Велика кількість альтернативних компаній, що пропонують подібний сервіс. Зміна провайдеру чи укладення нового контракту або підписки в іншому місці може зайняти лічені хвилини, а це означає, що компанія може втратити клієнтів так само швидко, як і здобути їх.

1.5 Існуючі підходи до прогнозування відтоку

З метою зменшення втрат клієнтів та збереження їх на довгострокову перспективу, компанії використовують різноманітні методи та підходи для прогнозування відтоку. До найпоширеніших належать.

1. Когортний аналіз, що є методом дослідження поведінки групи людей (когорти), які поділяють одну або декілька загальних характеристик, таких як час покупки, вік, місце проживання тощо. Основною метою когортного аналізу є вивчення тенденцій і змін у поведінці різних груп клієнтів з часом, зокрема зміни у кількості та частоті покупок, середньому чеку, відсотку повторних покупок, а також вивчення факторів, які впливають на ці зміни. Когортний аналіз може бути корисним для розуміння, які покращення повинні бути внесені в послуги компанії, щоб підвищити лояльність клієнтів, а також для розробки спеціальних програм та послуг для окремої групи клієнтів[15].

Для проведення когортного аналізу часто застосовують Microsoft Excel та Google Таблиці. Приклад когортного аналізу в Google Аналітиці можна побачити на рисунку 1.8.



	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
All Users 391 users	**100.00%**	**1.53%**	**1.49%**	**0.96%**	**1.18%**	**0.00%**	**0.00%**
Jan 17, 2018 87 users	100.00%	2.30%	3.45%	2.30%	2.30%	0.00%	0.00%
Jan 18, 2018 85 users	100.00%	0.00%	0.00%	0.00%	1.18%	0.00%	
Jan 19, 2018 82 users	100.00%	2.44%	0.00%	1.22%	0.00%		
Jan 20, 2018 58 users	100.00%	1.72%	3.45%	0.00%			
Jan 21, 2018 23 users	100.00%	4.35%	0.00%				
Jan 22, 2018 56 users	100.00%	0.00%					

Рисунок 1.8 – Візуалізація когортного аналізу в Google Аналітиці

2. Аналіз тексту, наприклад коментарів та відгуків клієнтів, які можуть свідчити про недоліки в роботі компанії та відповідно незадоволення клієнтів послугами. Інструменти, які використовуються, можуть бути різні, зокрема: сентимент-аналіз, що допомагає визначати емоційне забарвлення тексту; веб-скрапінг, тобто процес автоматичного збору даних з інтернет-сайтів; тематичний аналіз, що допомагає визначити головні теми та ключові слова і може бути корисним для виявлення основних проблем або потреб клієнтів. Тож, аналіз тексту дозволяє виявити проблеми, що спричиняють відтік клієнтів, та вчасно вжити заходів для їх вирішення.

3. Використання машинного навчання, що стає все більш популярним підходом до вирішення задачі прогнозування відтоку користувачів особливо в телекомунікаційній галузі. Це пояснюється тим, що застосування алгоритмів машинного навчання дозволяє автоматизувати процес аналізу великих обсягів даних, що робить його більш ефективним та точним.

Сучасні дослідники, як правило, використовують ансамблеві та гібридні методи машинного навчання для прогнозування відтоку.

У дослідженні 2022 року [16] Yana Fareniuk, Tetiana Zatonatska та інші провели порівняння різних моделей машинного навчання для прогнозування відтоку користувачів на основі даних однієї з провідних українських телекомунікаційних компаній, датасет містив історію взаємодії користувачів з мережею, їх профіль, тарифні плани, обсяги користування послугами оператора тощо. На етапі моделювання були впроваджені основні моделі (C5.0, KNN, Neural Net, Ensemble, Random Tree, Neural Net Ensemble) за допомогою IBM SPSS Modeler. Оцінка точності результатів досягла 95%, що свідчить про можливість та доцільність використання моделей в подальшій класифікації клієнтів з метою прогнозування та мінімізації відтоку споживачів.

Vijaya та Sivasankar [17] запропонували свій підхід до розв'язання поставленої задачі, вони використали Rough Set Theory (RST) для визначення факторів, які найбільше впливають на відтік клієнтів у телекомунікаційному секторі, застосування RST дозволило зменшити кількість незначущих ознак навчального набору даних та підвищити ефективність випробуваних моделей. Далі Vijaya та Sivasankar використовували методи ансамблевої класифікації, такі як Bagging, Boosting та Random Subspace. Для оцінки методів були використані такі метрики, як specificity, precision та accuracy, і визначено, що запропоноване рішення досить непогано виконує поставлену задачу з accuracy 94,13%.

Yahaya та інші провели дослідження [18], які показали негативний вплив дисбалансу класів на результати прогнозування, вони розробили алгоритм прогнозування відтоку, використовуючи моделі GA, K-means та ANN. Результати показали, що ефективність навчання покращується зі зменшенням шуму в даних, проте результати тестування не були покращені за допомогою фільтрації. Точність їхньої гібридної моделі (GA-K-means-ANN) впала з 97% на навчальних даних до 76,6% на тестових. Це пояснюється тим, що дисбаланс між позитивним та негативним класами значною мірою впливає на результати тестування.

1.6 Висновки до розділу 1

Отже, телекомунікації - важлива складова сучасного світу, а бізнес, який забезпечує нас цими послугами потребує підтримки. Для телекомунікаційних компаній важливо мати інструменти для аналізу і попередження відтоку аудиторії. Зупинити відтік користувачів неможливо. Але повна відмова від послуг оператора з боку клієнта не відбувається за один день, скоріше незадоволеність клієнта, що зростає з часом, посилюється відсутністю уваги з боку провайдера та призводить до втрати споживачів. Якщо телекомунікаційна компанія може ідентифікувати клієнтів, які, ймовірно, розірвуть співпрацю з оператором, то вона може запропонувати їм конкретні рішення, щоб зменшити їхнє незадоволення, підвищити залученість і, таким чином, потенційно утримати їх, це ключ до вдосконалення та еволюції сервісу.

Оскільки витрати на залучення нових клієнтів є відносно високими, телекомунікаційні компанії сьогодні зосереджуються на стратегії мінімізації відтоку, шляхом утримання своїх довгострокових клієнтів, адже утримувати аудиторію - вигідно. Враховуючи вищезазначене, утримання клієнтів є вимогою часу, і розуміння відтоку є важливим для досягнення цієї мети, тож в цій роботі розглянуто проблему прогнозування відтоку клієнтів як проблему бінарної класифікації, в якій всі клієнти поділяються на два класи: тих, що залишаються з компанією і тих, що на межі відтоку. А машинне навчання є ефективним інструментом для вирішення поставленої задачі.

2 АНАЛІЗ МЕТОДІВ МАШИННОГО ДЛЯ ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ЗАВДАННЯ КЛАСИФІКАЦІЇ

В першому розділі було досліджено вплив проблеми відтоку аудиторії на телеком індустрію та на бізнес в цілому. Визначено актуальність передбачення рівня відтоку та розуміння його причин з метою покращення обслуговування, що надається бізнесом. Тепер час зосередитися на практичних аспектах вирішення цієї задачі. Ефективним інструментом, який був обраний для прогнозування відтоку в даній роботі, є машинне навчання, зокрема, такі методи машинного навчання як логістична регресія, дерева прийняття рішень, наївна баєсівська модель, метод опорних векторів та градієнтний бустинг. Тож, в даному розділі будуть розглянуті усі перераховані вище моделі, їх математичний апарат та переваги їх використання для вирішення поставленої в минулому розділі задачі класифікації.

2.1 Машинне навчання: від теорії до практики

Історія розвитку машинного навчання бере початок у 1950 році, з минулих десятиліть загальний принцип не змінився, проте багаторазово ускладнилися закономірності, які вони здатні виявляти та прогнози, які вони створюють, розширилося коло задач, які розв'язуються з їх використанням, це в свою чергу зумовлено стрімким зростанням обчислювальних потужностей сучасних комп'ютерів. Ринок машинного навчання швидко зростає. З 2016 року його обсяг досяг позначки $1 млрд, а за прогнозами, до 2025 року він може збільшитися до $39,98 млрд. Від виробництва до роздрібної торгівлі, від банківської справи до системи охорони здоров’я, навіть традиційні компанії використовують його як інструмент для вирішення

складних проблем, оптимізації багатьох процесів, підвищення ефективності бізнесу та стимулювання зростання в цілому.

Машинне навчання - це напрям штучного інтелекту, що вивчає методи побудови алгоритмів, здатних навчатися. В його основі лежить ідея про те, що системи можуть навчатися на даних, виявляти закономірності і кореляції у великих наборах даних, а також приймати оптимальне рішення і створювати прогнози на основі проведеного аналізу з мінімальним втручанням людини. Завдяки адаптивному характеру машинне навчання доцільно використовувати в задачах, де дані є не повними, спотвореними, не систематизованими або постійно змінюються. Тож, це особливо актуально в телеком індустрії, де кожного дня оператори, що надають послуги зв'язку, обробляють величезні об'єми даних, якість даних може бути різною, виявляти закономірності між ними стає все складніше, а машинне навчання може допомогти частково чи повністю автоматизувати рішення складних аналітичних задач, з якими компанії зустрічаються кожного дня. З його допомогою можна досягати максимально точних прогнозів на основі вхідних даних, щоб бізнес-власники, маркетологи та інші працівники могли приймати правильні рішення у своїй роботі. Після проходження навчання, моделі здатнм передбачати результат, відтворювати за потреби, запам'ятовувати його та вибирати оптимальний варіант серед декількох, а результати стають точнішими зі збільшенням обсягу даних, що проходять обробку моделлю, а також з тривалішим її використанням[19].

Алгоритми машинного навчання можна умовно розділити на три великі класи в залежності від досвіду та набору даних для навчання, а саме.

1. Навчання з учителем (supervised learning). Цей підхід заснований на використанні маркованих навчальних даних. Маркування даних - це процес категоризації вхідних даних з відповідними їм певними вихідними значеннями. Моделі надаються як вхідні, так і вихідні дані, вона навчається на прикладах надавати правильну відповідь, задану «вчителем», порівнюючи свої фактичні результати з правильними, щоб знайти помилки, потім

відповідно модифікує модель. Щоб отримувати достатньо точні результати навчальні дані мають бути схожими на дані, на яких потім буде застосовуватися побудована модель.

2. Навчання без учителя (unsupervised learning). Під час навчання без учителя модель навчається на немаркованих даних, вона сканує набори даних у пошуках будь-якого значущого зв'язку, визначає кореляції, намагається виявити шаблони. Після цього, алгоритм намагається впорядкувати ці дані, щоб описати їх структуру.

Навчання без вчителя використовується для задач кластеризації, коли потрібно розділити дані на наперед невідомі класи, використовуючи деяку міру схожості, наприклад, персоналізація користувача веб-сайту, сегментація медичного знімку для виявлення захворювання; для задач оцінки щільності, коли необхідно оцінити розподіл, з якого отримано вхідні дані, знаючи апріорні ймовірності їх появи; для задач синтезу або очищення від шуму[20].

3. Навчання з частковим залученням учителя (semi-supervised). Як випливає з назви, цей метод поєднує в собі навчання з учителем і без нього. Він заснований на використанні невеликої кількості розмічених даних і великої кількості нерозмічених даних для навчання. Модель спочатку навчається на нерозмічених даних, а потім, використовуючи це наближення, донавчається на розмічених[20]. Навчання з частковим залученням учителя може вирішити проблему недостатньої кількості маркованих даних для алгоритму навчання з учителем.

Саме до навчання з учителем відноситься поставлена в даній роботі задача класифікації. Навчальні дані, на яких тренуватимуться згадані вище моделі, будуть промарковані, а саме, для кожного прикладу з навчальної вибірки буде вказаний відповідний клас (churn – ті користувачі, що на межі відтоку; no churn – та частина аудиторії, що продовжує активну взаємодію з компанією). Після навчання на цих даних модель має самостійно класифікувати приклади, класи яких невідомі.

Тож, машинне навчання включає різні типи моделей із застосуванням різних алгоритмічних методів. Який алгоритм працює найкраще залежить від типу розв'язуваного завдання, доступних обчислювальних ресурсів, характеру даних тощо, їх можна використовувати по одному або комбінувати для досягнення максимально можливої точності. Але процес машинного навчання можна представити в кроках, що зображені на рисунку 2.1.

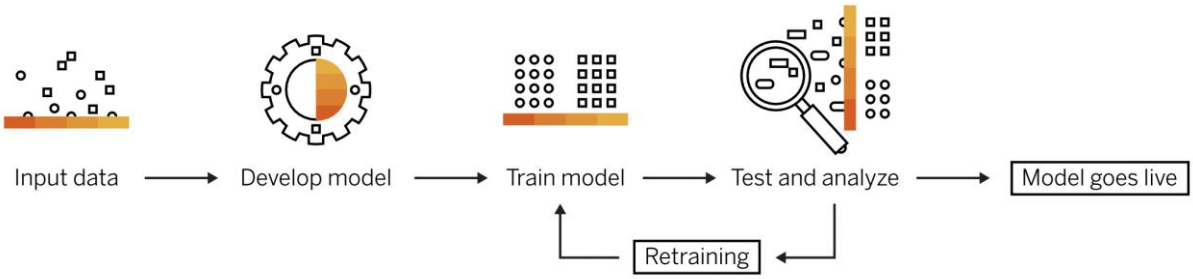


Рисунок 2.1 – Процес машинного навчання

2.2 Постановка задачі класифікації

Класифікація є важливою і досить поширеною задачею машинного навчання, це процес розбиття об'єктів заданого набору даних на групи (класи), що є апріорно відомими, на основі аналізу їхнього формального опису. Під час класифікації кожен об'єкт належить до певної групи (класу) або номінальної категорії на основі деякої якісної властивості.

У машинному навчанні задачу класифікації розв'язують із використанням навчання з учителем, оскільки класи визначають заздалегідь і для об’єктів навчальної вибірки мітки класів задано. Аналітичні моделі, що використовуються для розв’язання задачі класифікації, називаються класифікаторами[21].

Як вже було згадано вище, для проведення класифікації необхідно мати формальний опис об'єкта, з яким можна працювати, використовуючи

математичний апарат класифікації. Найчастіше таким описом є база даних, кожен запис якої містить інформацію про певну властивість об'єкта. В межах заданої предметної області властивостями об’єкта є інформація про користувачів, наприклад, їх персональні дані чи обсяги користування послугами оператора (доступ до мережі Інтернет, дзвінки, SMS-повідомлення тощо).

В процесі класифікації можна виділити декілька етапів.

1. Набір вхідних даних розподіляють на навчальну (training set) і тестову (test set) вибірки. Для досягнення високої точності класифікатора, який буде побудований далі, кількість об'єктів у навчальній вибірці має бути досить великою, вона має містити об'єкти, що представляють усі можливі класи. Тестовий набір даних також має містити вхідні та вихідні значення параметрів, адже останні будуть використовуватися для оцінки ефективноті моделі.

2. На основі навчальної вибірки, що містить об'єкти для яких відомі значення залежних та незалежних змінних, відбувається побудова моделі класифікації.

3. Оцінка точності класифікатора. На цьому етапі перевіряється наскільки точно побудована модель визначає класи для нових об'єктів, які не входять до навчальної вибірки.

При вирішенні задач класифікації можуть виникати різні проблеми, зокрема незадовільна якість вхідних даних, які можуть містити пропущені чи помилкові значення, шуми, тобто непередбачувані аномальні значення в даних, які не відповідають загальному шаблону і можуть спотворювати аналіз та прогнозування, різні типи даних, різна значимість атрибутів, дані можуть бути незбалансовані (тобто кількість навчальних прикладів, представників позитивного і негативного класів, дуже різниться), що також може вплинути на якість побудованої моделі. Крім того, можна зіткнутися з такими явищами як overfitting чи underfitting.

Overfitting - це проблема в машинному навчанні, коли модель надто добре підлаштовується під навчальний датасет і неправильно прогнозує дані,

які не були використані під час її тренування. Класифікатор надмірно запам'ятовує залежності в тренувальному датасеті, намагаючись інтерпретувати помилки, шуми та випадкові залежності як частину внутрішньої структури даних, що призводить до поганої ефективності моделі на тестовому датасеті, де характер помилок може бути іншим.

Проблема underfitting полягає в тому, що модель недостатньо вивчила характеристики даних в навчальному наборі, тому вона не може виконувати правильні прогнози на нових даних. Це може статися через недостатню складність моделі, недостатню кількість навчальних даних, або через невдалі налаштування параметрів моделі. У результаті, класифікатор може недооцінювати складність даних та/або не виявляти важливі закономірності, недооцінювати значимість деяких атрибутів, які є важливими для правильних передбачень, все це призводить до низької точності моделі.

Тож, далі розглянемо детальніше ті моделі, що були обрані для розв’язання задачі класифікацій в даній роботі.

2.3 Логістична регресія

Логістична регресія - це алгоритм машинного навчання, який часто використовується для розв'язання задач бінарної класифікації. Вона отримала свою назву завдяки тому, що використовує логістичну функцію для прогнозування ймовірності належності об'єкта до одного з класів. Логістична регресія використовує лінійну комбінацію вхідних ознак і відповідних ваг, яка описує лінійну гіперплощину в просторі ознак. Потім цей результат проходить через логістичну функцію, яка переводить лінійну комбінацію в імовірність належності об'єкта до одного з класів.

Усі регресійні моделі можуть бути записані в наступному вигляді:

$$y = F(x_1, x_2, \dots x_n)$$

У множинній лінійній регресії вважається, що залежна змінна є лінійною функцією від незалежних змінних, тобто:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$$

Виникає питання, чи можна її використовувати для задачі оцінки ймовірності результату події? Відповідь є позитивною, ми можемо обчислити коефіцієнти регресії, однак виникає проблема: множинна лінійна регресія не враховує той факт, що залежна змінна є бінарною. Це призводить до створення моделі з передбачуваними значеннями, які можуть бути більшими за 1 або меншими за 0, але такі значення є неприпустимими для поставленої задачі. Таким чином, множинна лінійна регресія просто не враховує обмеження на діапазон значень для залежної змінної.

Для вирішення цієї проблеми задача регресії може бути сформульована інакше: замість передбачення бінарної змінної, ми передбачаємо неперервну змінну зі значеннями на відрізку [0,1] за будь-яких значень незалежних змінних. Це досягається застосуванням наступного регресійного рівняння (логіт-перетворення):

$$\hat{p} = \frac{1}{1 + \exp(-y)} \qquad (2.1)$$

Подібно до лінійної регресійної моделі, логістична регресійна модель підраховує зважені суми вхідних ознак, але замість видачі результату безпосередньо, як це робить лінійна регресійна модель, вона видає логіт-перетворення результату.

Логіт-перетворення, представляє собою сигмоідальну (тобто S – подібної форми) функцію, що видає значення між 0 та 1. Вона визначена як показано в рівнянні (2.1) та на рисунку 2.2.

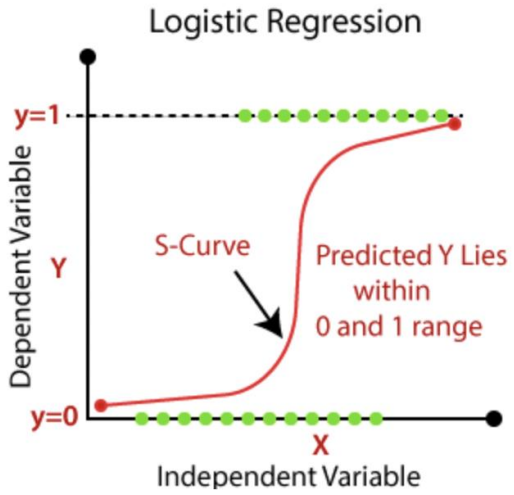


Рисунок 2.2 – Логістична крива

Після того як логістична регресійна модель оцінила ймовірність $\hat{p}$, що приклад x належить певному класу, вона може дати свій прогноз:

$$\hat{y} = \begin{cases} 0, \text{якщо } \hat{p} < 0.5 \\ 1, \text{якщо } \hat{p} \geq 0.5 \end{cases}$$

Варто зазначити, що $\hat{p} < 0.5$, коли $y < 0$, а $\hat{p} \geq 0$, коли $y \geq 0$, тож логістична регресійна модель прогнозує 1, якщо $b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$ набуває додатніх значень та 0, якщо від'ємних.

Розглянемо детальніше визначення логістичної моделі[22]. Припустимо, що маємо деяку випадкову величину Y, вона може приймати тільки значення 0 та 1, а також залежить від деякої множини пояснювальних змінних $x = (1, x_1, x_2 \dots x_n)^T$. Введемо додаткову змінну y^* для визначення залежності Y від $x_1, x_2 \dots x_n$, маємо:

$$y^* = \theta^T x = \theta_0 + \theta_1 x_1 + \cdots \theta_n x_n + \varepsilon$$

Тоді:

$$Y = \begin{cases} 0, & y^* \leq 0 \\ 1, & y^* > 0 \end{cases}$$

Доданок ε вважаємо випадковою величиною, що має логістичний розподіл. Отже, для конкретних значень пояснювальних змінних $x^* = x_1^*, x_2^* \dots x_n^*$ маємо відповідні значення y^*, а ймовірніть того, що Y набуває значення 1 наступна:

$$p(Y = 1) = p(y^* > 0) = p(\theta^T x^* + \varepsilon > 0) = p(\varepsilon > -\theta^T x^*) =$$
$$= p(\varepsilon \leq \theta^T x^*) = \Lambda(\theta^T x^*)$$

Тут, $\Lambda -$ функція розподілу для логістичного розподілу відповідно, та має вигляд $\Lambda(x) = \frac{1}{1+e^{-x}}$, а рівність $p(\varepsilon > -\theta^T x^*) = p(\varepsilon \leq \theta^T x^*)$ випливає з симетричності логістичного розподілу.

Отже, маємо, що для певного значення $\mathrm{x^i}$ відповідна випадкова величина $\mathrm{Y^i}$ має розподіл Бернулі: $\mathrm{Y^i \sim B(1, \theta^T x^i))}$.

Умова, що є основною для побудови логіт-моделі:

$$\ln \frac{\mathrm{p(1|X)}}{1 - \mathrm{p(1|X)}} = \ln \frac{\mathrm{p(1|X)}}{\mathrm{p(0|X)}} = \mathrm{b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n}$$

Оцінка параметрів $\mathrm{\theta_0, \theta_1 \dots \theta_n}$ на основі вибірки $\mathrm{(x^1, Y^1) \dots (x^m, Y^m)}$, де $\mathrm{x^i \in R^n -}$ певні значення незалежних змінних, $\mathrm{Y^i \in \{0,1\} -}$ відповідні значення залежної змінної для кожного прикладу, здійснюється з допомогою ММП (метод максимальної правдоподібності). Згідно ММП в якості оцінки параметра приймаються такі значення θ, що забезпечують максимізацію функції правдоподібності L:

$$\hat{\theta} = \mathrm{argmax_\theta L(\theta) = argmax_\theta \prod_{i=1}^{m} Pr\{Y = Y^i | x = x^i\}}$$

Пошук оцінки спрощується, якщо максимізувати не саму функцію L, а натуральний логарифм від неї, оскільки максимум обох функцій досягається при однакових значеннях θ:

$$\mathrm{lnL(\theta) = \sum_{i=1}^{m} lnPr\{Y = Y^i | x = x^i\} =}$$

$$= \sum_{i=1}^{m} \mathrm{Y^i ln\Lambda(\theta^T x^i) + (1 - Y^i) ln(1 - \Lambda(\theta^T x^i)}$$

Логарифмічна функція правдоподібності всюди увігнута, тож для пошуку максимуму можна використати метод Ньютона, який в даному випадку буде завжди сходитися, адже виконано умову збіжності, також можна застосувати й інші алгоритми оптимізації, наприклад градієнтний спуск або стохастичний градієнтний спуск.

Переваги логістичної регресії.

1. Логістична регресія показує хорошу масштабованість, що означає, що вона може бути застосована до великих наборів даних без великої втрати продуктивності.

2. Логістична регресія є простим та зрозумілим алгоритмом, що дозволяє легко інтерпретувати результати.

3. Логістична регресія не вимагає, щоб вхідні змінні були незалежними одна від одної. Вона може працювати зі змінними, які взаємозв'язані або корельовані.

4. Логістична регресія надає ймовірнісну інтерпретацію результатів, дозволяючи оцінити ймовірність віднесення до певного класу.

5. Невелика схильність до перенавчання, оскільки вона не має безлічі параметрів, які потрібно оптимізувати.

Недоліки логістичної регресії.

1. Логістична регресія вимагає нормалізації ознак, щоб гарантувати однаковий внесок ознак у модель.

2. Погано працює на даних з великою кількістю ознак або складною структурою даних.

3. Логістична регресія може давати низьку точність, якщо класи не є лінійно роздільними.

2.4 Дерева рішень

Дерева рішень є одним із найефективніших інструментів інтелектуального аналізу даних, який використовують для розв'язання задач класифікації та регресії.

Основоположні ідеї, що сприяли появі та подальшому розвитку алгоритму, були закладені в 1950-х роках у галузі досліджень моделювання поведінки людини комп'ютерними системами. Роботи Е. Ханта та ін. "Експерименти з індукції" [23] та "Комп'ютерне моделювання мислення" К. Ховеленда [24] відіграли важливу роль у цьому процесі. На подальший розвиток моделі вплинули також роботи Лео Бреймана [25], який запропонував метод випадкового лісу і алгоритм CART, а також Джона Р. Квінлена [26], який розробив алгоритм ID3 і його модифікації C4.5, C5.0.

Дерева рішень - це метод представлення правил у вигляді ієрархічної послідовної структури, де кожному об'єкту відповідає окремий вузол, що приймає рішення. Правило у дереві рішень виражається як логічна конструкція виду "якщо ... то ...". Так як воно сформульовано практично природною мовою, дерева рішень як аналітичні моделі є більш інтерпретованими та вербалізованими, ніж, наприклад, нейронні мережі. Ця властивість корисна при інтерпретації моделі класифікації в цілому, адже дозволяє створювати класифікатори в тих сферах, де досліднику досить складно формалізувати знання.

Результуюча структура під час візуалізації має вигляд дерева з різними типами вузлів, а саме зв’язний граф з множиною вузлів, який не містить циклів і має окремий вузол, в яку не входить жодне ребро, він називається коренем дерева, два інші види вузлів – внутрішні вузли прийняття рішень та листи. Корінь є відправною точкою для дерева рішень, який потім розгалужується на внутрішні вузли та листи. У вузлах містяться вирішальні правила (decision

rules) і проводиться перевірка відповідності прикладів цьому правилу за певним атрибутом навчальної множини.

У найпростішому випадку, множина прикладів, що потрапили до вузла, розбивається на дві підмножини, в одну з яких потрапляють приклади, що задовольняють правило, а в іншу - ті, що ні. Далі процес рекурсивно повторюється на кожній підмножині, доки не виконається умова зупинки алгоритму. Зрештою в останньому вузлі перевірку і розбиття не проводять, і його оголошують листом. Лист визначає рішення для кожного прикладу, що потрапив у нього, у випадку класифікації - це клас, асоційований із вузлом. Отже, лист містить підмножину об'єктів, для яких виконуються усі правила відповідної гілки, а оскільки шлях до кожного листа дерева є єдиним, то і кожен приклад може бути віднесений лише до одного листа, що гарантує єдиність розв'язку. На рисунку 2.3 зображена структура дерева[28]:

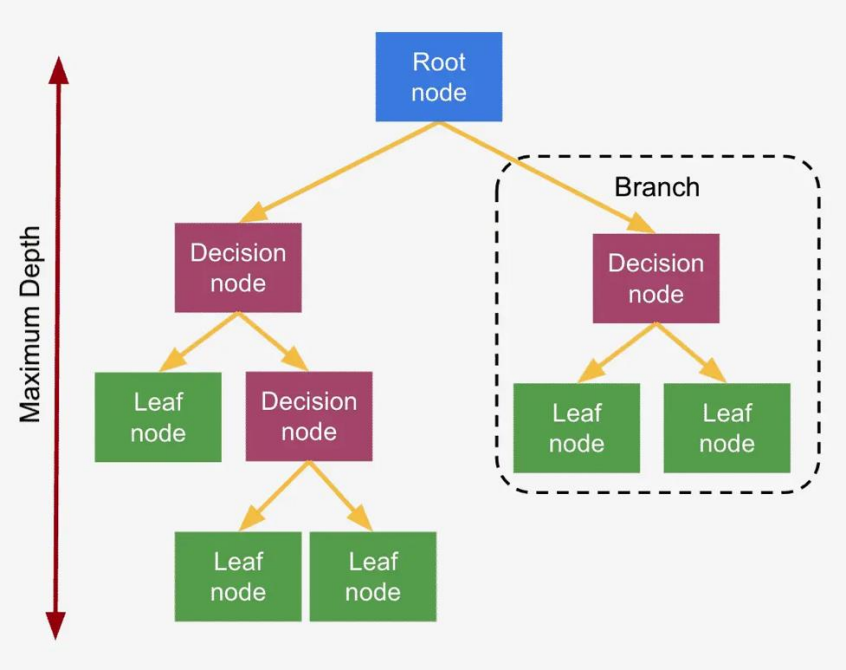


Рисунок 2.3 – Структура дерева рішень

Так як правила в деревах рішень формуються в наслідок узагальнення великої кількості навчальних прикладів, що описують предметну область, то за аналогією з відповідним методом логічного виведення вони отримали назву

- індуктивні правила, а процес створення дерев рішень на основі цих правил - індуктивне навчання.

Кожному прикладу з навчальної вибірки повинна відповідати заздалегідь задана залежна змінна, оскільки дерева рішень відносяться до методів навчання з учителем. До того ж, якщо цільова змінна є неперервною, модель вважається деревом регресії, а коли вона є дискретною, то деревом класифікації.

Розглянемо трохи детальніше процес побудови[29]. Дерева рішень будуються шляхом послідовного та рекурсивного розбиття навчальної множини на підмножини з використанням вирішальних правил у вузлах. Цей процес триває до тих пір, поки всі вузли на кінці гілок не стануть листками. Вузол може бути оголошений листом, якщо він включає лише один об'єкт або об'єкти, що є представниками одного класу, а також у випадку виконання певної умови зупинки алгоритму, яка встановлюється дослідником (наприклад, досягнення максимальної глибини дерева або встановлення мінімальної допустимої кількість прикладів у вузлі). Алгоритми побудови дерев рішень відносяться до класу жадібних алгоритмів. Таку назву отримали алгоритми, які припускають, що прийняття локально-оптимальних рішень на кожному кроці (розбиття у вузлах), призведе до глобально оптимального рішення. Тобто в контексті алгоритму, якщо атрибут розбиття один раз вже було обрано, то повернутися назад і обрати інший, який ймовірно міг би забезпечити краще розбиття, вже неможливо. Тож на етапі побудови дерева не можна стверджувати напевно, чи зможе обраний атрибут в результаті забезпечити оптимальне розбиття.

S – множина, що задана для навчання та містить n прикладів, для кожного відома мітка класу $C_i (i = 1 \dots k)$ та r атрибутів $A_j (j = 1 \dots r)$, які, як припускається, визначають до якого класу належить приклад. Тоді, можливі наступні випадки.

1. Множина S містить приклади, що відносяться тільки до одного класу C_i. Зрозуміло, що в такому випадку навчання не матиме сенсу, так як всі

приклади, що подаються моделі, будуть належати одному класу, який вона і «навчиться» розпізнавати. Тож, дерево рішень буде являти собою лист, асоційований із класом C_i, практичне використання якого беззмістовне, оскільки будь який новий приклад така модель буде відносити саме до цього класу.

2. Множина S є порожньою множиною. В такому випадку для нього все ж буде створено лист, але клас буде обраний з іншої множини (це може бути клас, який виявиться найбільш частим в батьківській множині)

3. Множина S складається з навчальних прикладів, що є представниками усіх класів C_i. В такому випадку необхідно розбити множину S на підмножини, асоційовані з відповідними класами. З цією метою обирається один з атрибутів A_j множини D, що містить унікальні значення $(a_1, a_2 \dots a_p)$, де $p \geq 2$ – кількість унікальних значень атрибуту. Далі множина S розбивається на p підмножин $(S_1, S_2 \dots S_p)$, кожна з яких включає приклади, що містять відповідні значення атрибуту. Далі обирається інший атрибут і процедура рекурсивно повторюється, доки усі приклади в результуючих підмножинах не належатимуть до одного класу.

Описаний процес є основою багатьох сучасних алгоритмів побудови дерев рішень. Очевидно, що під час застосування даної методики, дерево будується згори вниз, починаючи з кореня і закінчуючи листям.

Під час побудови дерева рішень потрібно розв'язати кілька важливих проблем, а саме.

1. Вибір атрибута, за яким буде здійснюватися розбиття.

При формуванні чергового правила потрібно обрати атрибут, відносно якого у вузлі буде виконано розбиття. Він має поділити множину спостережень таким чином, щоб підмножини складались з прикладів одного класу, або були близькі до цього. Іншими словами, кількість об'єктів з інших класів, так званих "домішок", в кожній з цих підмножин повинна бути мінімальною. Для цього використовують різні критерії (ентропійні, хі-

квадрат, джині), найпопулярнішими з яких стали статистичний та теоретико-інформаційний.

Теоретико-інформаційний критерій, як можна здогадатися з його назви, ґрунтується на основних поняттях теорії інформації, зокрема на інформаційній ентропії.

$$H = -\sum_{i=1}^{n} \frac{N_i}{N} \log\left(\frac{N_i}{N}\right)$$

Тут N – загальна кількість прикладів в підмножині, n – кількість класів у вихідній підмножині, N_i – к-сть прикладів і-го класу.

Отже, ентропія може бути розглянута як показник неоднорідності класів, представлених у підмножині. Якщо класи представлені в рівних відношеннях і невизначеність класифікації висока, ентропія набуває максимального значення. У випадку, коли всі приклади вузла належать одному класу (тобто $N = N_i$) , ентропія набуває нульового значення, оскільки логарифм від одиниці дорівнює нулю. Отже, необхідно обрати такий атрибут A_j, що забезпечить мінімальне значення ентропії результуючої підмножини відносно батьківської. Варто згадати ще одне поняття, адже на практиці використовують не саму ентропію, а її обернену величину, яку називають інформацією. В такому випадку найкращий атрибут має забезпечить максимальний приріст інформації результуючого вузла відносно вихідного:

$$\mathrm{Gain}(A) = \mathrm{Info}(S) - \mathrm{Info}(S_A)$$

Тут, $\mathrm{Info}(S)$ – інформація, що пов’язана з підмножиною S до здійснення розбиття, $\mathrm{Info}(S_A)$ – інформація, що пов’язана з підмножиною, отриманою після розбитті за атрибутом A.

В основі статистичного підходу лежить використання індексу Джині. Статистичний зміст цього показника полягає в тому, що він вказує, наскільки часто випадково обраний приклад навчальної множини буде класифіковано неправильно, якщо цільові змінні мають певний статистичний розподіл. Тобто, значення індексу Джині вказує на відстань між двома розподілами -

розподілом цільових змінних і розподілом передбачень моделі, чим менше це значення, тим кращою вважається модель.

Формула розрахунку індексу Джині:

$$\mathrm{Gini}(Q) = 1 - \sum_{i=1}^{n} p_i^2,$$

Тут Q – результуюча множина, n – кількість класів у ній, p_i– ймовірність і-го класе. Показник належить проміжку від 0 до 1, набуває значення 0, якщо всі приклади результуючої множини є представниками одного класу, а значення 1, якщо класи представлені в однакових пропорціях.

2. Критерій зупинки алгоритму

В теорії, навчання дерева рішень може тривати аж до моменту створення таких підмножин, що міститимуть представників виключно одного класу. Проте, в результаті ми можемо отримати дерево, в якому кожному прикладу відповідатиме окремий лист. Таке дерево є перенавченим, а структура складною, його побудова і подальше використання є недоцільним, оскільки кожен приклад буде мати власний унікальний шлях та набір правил, які застосовуються тільки до нього. З метою уникнення такої ситуації варто заздалегідь визначити критерій зупинки алгоритму.

Один із підходів - рання зупинка. Цей метод передбачає зупинку алгоритму, якщо досягнута задана метрика або вимога, наприклад, певний відсоток правильно розпізнаних прикладів. Це може значно скоротити час навчання, але скоріш за все точність дерева буде страждати в результаті такої зупинки.

Інший підхід - обмеження глибини дерева. Навчання зупиняється після досягнення заздалегідь встановленого максимального числа розбиттів у гілках. Хоча цей метод може знизити точність дерева, він допомагає уникнути перенавчання та складності структури.

Третій підхід - встановлення мінімально допустимої кількості прикладів у вузлі. Це дозволяє уникнути непотрібних розбиттів та створення незначущих правил.

Всі ці підходи є евристичними і не гарантують найкращого результату, а вибір критерія зупинки залежить від поставленої задачі, набору даних, тощо.

3. Відсікання гілок

Розростання дерева, як було зазначено вище, не обмежити. Але скоротити складність дерева можна шляхом видалення непотрібних гілок, які є малозначущими з практичної точки зору. Це є ефективним підходом для досягнення балансу між складністю і точністю дерева.

Процес відсікання гілок розпочинається з побудови повного дерева, де кожен листок містить приклади одного класу. Далі, визначаються два показники - відносна точність моделі, яка відображає співвідношення правильно розпізнаних прикладів до загальної кількості, і абсолютна помилка, що відображає кількість неправильно класифікованих прикладів.

Після цього, здійснюється відсікання гілок, шляхом послідовного перетворення вузлів на листки, що не суттєво впливають на точність моделі або можуть збільшити помилку. Цей процес здійснюється знизу вгору, протилежно напрямку його зростання.

Варто зауважити, що відсікання гілок може зайняти більше часу ніж рання зупинка, оскільки перед самим відсіканням необхідно побудувати повне дерево. Проте, цей додатковий час навчання компенсується кращою здатністю дерева до узагальнення і зменшенням ризику перенавчання.

Окрім відсікання гілок, існує ще один підхід до скорочення складності дерева рішень - вилучення правил. Навіть після виконання відсікання гілок, дерево може залишатися складним для візуального сприйняття та інтерпретації. В такому випадку, можна вилучити вирішальні правила з дерева і організувати їх у набори, які описують класи.

Для вилучення правил, необхідно прослідкувати всі шляхи від кореневого вузла до листків дерева. Кожен шлях відображає вирішальне правило, що складається з умов, які перевіряються на в кожному вузлі. Цей підхід дозволяє представити складне дерево у вигляді зрозумілих вирішальних правил.

На останок, сьогодні існує значна кількість алгоритмів для навчання дерева рішень, наприклад CART, ID3, C4.5, C5.0, CHAID, CN2 тощо. Проте, особливої популярності здобули такі.

1. ID3 (Iterative Dichotomizer 3) – алгоритм спеціалізується на роботі з дискретною цільовою змінною, тому дерева рішень, побудовані з його використанням, є класифікаційними. Кількість нащадків в кожному вузлі дерева є необмеженою. Не може працювати з пропущеними даними. Варто відмітити, що ID3 не працює з пропущеними даними. Вибір атрибуту відбувається на основі приросту інформації (Gain) або критерію Джині.

2. C4.5 - модифікована версія алгоритму ID3, яка дає можливість працювати з пропущеними значеннями. Вибір атрибуту відбувається на основі нормалізованого приросту інформації (Gain Ratio).

3. CART (Classification and Regression Tree) - алгоритм, що дає змогу розв'язувати задачі класифікації і регресії, тобто використовувати як дискретну, так і неперервну цільову змінну. Алгоритм будує дерева, які в кожному вузлі мають тільки два нащадки.

2.5 Наївний баєсівський класифікатор

Ще одним методом машинного навчання, який варто розглянути для вирішення задачі прогнозування відтоку в телеком індустрії, є наївний баєсівський класифікатор.

Це алгоритм навчання з учителем, принцип роботи якого базується на застосуванні теореми Баєса з «наївним» припущенням про незалежність ознак, які характеризують приклад з навчального набору даних. У реальних даних часто виникає кореляція між ознаками, і це припущення може бути недостатнім для точної класифікації. Проте це спрощення дозволяє зменшити обчислювальну складність алгоритму та спростити процес моделювання.

Хоча модель і виходить досить простою, а припущення «наївними», класифікатор показує хороші результати в реальних практичних задачах. До того ж дана модель машинного навчання не вимагає великої кількості навчальних даних для його побудови, що несумнівно є її перевагою.

Нехай задана множина прикладів, кожен з яких має $x_1, x_2 \dots x_n$ ознак. Ймовірнісна модель класіфікатора – це умовна модель $p = (C_k|x_1 \dots x_n)$.

[30] Основна проблема полягає в тому, що коли кількість ознак n дуже велика, до того ж вони можуть набувати різних значень, тоді будувати таку модель на ймовірнісних таблицях стає неможливим. Тож, варто переформулювати модель, щоб вона легко піддавалась обробці. Використовуючи Теорему Баєса, маємо:

$$p(C_k|x_1 \dots x_n) = \frac{p(C_k)p(x_1 \dots x_n|C_k)}{p(x_1 \dots x_n)} \qquad (2.2)$$

На практиці нас цікавить тільки чисельник цього дробу, адже знаменник не залежить від класу C_k, значення $x_1 \dots x_n -$ відомі, тож знаменник є константою. Чисельник еквівалентний сумісній ймовірності моделі $p(C_k, x_1 \dots x_n)$, що може бути записана наступним чином:

$$\begin{aligned} p(C_k, x_1 \dots x_n) &= p(C_k)p(x_1 \dots x_n|C_k) = p(C_k)p(x_1|C_k)p(x_2 \dots x_n|C_k, x_1) \\ &= p(C_k)p(x_1|C_k)p(x_2|C_k, x_2)p(x_3 \dots x_n|C_k, x_1, x_2) \\ &= p(C_k)p(x_1|C_k)p(x_2|C_k, x_1) \dots p(x_n|C_k, x_1, x_2, x_3 \dots x_{n-1}) \end{aligned}$$

Тепер вводиться припущення, що атрибути x_i та x_j є умовно незалежними при $i \neq j$. А отже $p(x_i|C_k, x_j) = p(x_i|C_k)$. Таким чином, сумісна модель може бути виражена як (2.6).

$$\begin{aligned} p(C_k, x_1 \dots x_n) &= p(C_k)p(x_1|C_k)p(x_2|C_k)p(x_3|C_k) \dots p(x_n|C_k) \\ &= p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \end{aligned}$$

Тож, враховуючи попередні міркування, (2.2) приймає вигляд:

$$p(C_k|x_1 \dots x_n) = \frac{p(C_k) \prod_{i=1}^{n} p(x_i|C_k)}{p(x_1 \dots x_n)}$$

Наївний баєсівський класифікатор поєднує ймовірнісну модель, побудовану вище, з правилом прийняття рішень. Одне загальне правило повинно обрати найбільш ймовірну гіпотезу, воно відоме як апостеріорне правило прийняття рішення (MAP). Відповідний класифікатор, представляє собою функцію, що назначає мітку класа C_k для деякого k наступним чином:

$$\hat{y} = \operatorname{argmax} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

2.6 Метод опорних векторів

Не менш ефективним, ніж розглянуті вище моделі машинного навчання, є метод опорних векторів. Основним завданням алгоритму є пошук гіперплощини, що розділяє дані на два класи, в нашому випадку ту частину аудиторії, яку компанія втрачає і ту, яка підтримує активну взаємодію з нею.

Кожний приклад навчального набору даних належить одному з двох класів. Питання полягає в тому, чи можна розділити їх оптимальною гіперплощиною. Це є типовий випадок лінійної роздільності. Гіперплощин може бути багато, проте вважається, що максимізація зазору між класами сприяє більш точній класифікації. Тобто, найкращою гіперплощиною вважатиметься та, що забезпечить максимальну відстань від неї до найближчої точки. Якщо така площина існує, вона називається оптимальною розділюючою гіперплощиною.

Приклади навчальної вибірки представляються у вигляді точок, що мають наступний вигляд: $\{(x_1, c_1), (x_2, c_2) \dots (x_n, c_n)\}$, де $c_i -$ мітка класу відповідного прикладу, що приймає значення 1 або -1, $x_i - p$-вимірний вектор ознак прикладу, зазвичай нормалізований до значень [0,1]. Якщо не проводити нормалізацію, то точки з більшим відхиленням від середніх значень координат точок будуть мати значний вплив на сам класифікатор. Тож, в основі

алгоритма лежить процес побудови розділюючої гіперплощини, що має наступний вигляд, як зображено на рисунку 2.4 [31].

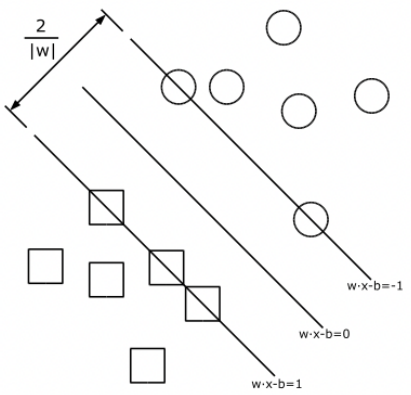


Рисунок 2.4 – Оптимальна розділююча гіперплощина

$w \cdot x - b = 0$, тут вектор w − перпендикуляр до розділюючої гіперплощини. Параметр $\frac{\mathrm{b}}{||\mathrm{w}||}$ рівний по модулю відстані від гіперплощини до початку координат. Тож, якщо параметр b дорівнює нулю, гіперплощина буде проходити через початок координат, що обмежуватиме розв'язок.

Оскільки нас цікавить оптимальне розбиття, нас цікавлять опорні вектори та гіперплощини, паралельні оптимальній і наближені до опорних векторів класів. Згадані гіперплощини можуть бути записані наступними рівняннями (2.3).

$$\begin{aligned} w \cdot x - b &= 1 \\ w \cdot x - b &= -1 \end{aligned} \tag{2.3}$$

Якщо навчальна вибірка є лінійно роздільною, то можна обрати гіперплощини таким чином, щоб між ними не було жодних точок і потім максимізувати відстань між ними. Ширина смуги між площинами рівна $\frac{2}{||\mathrm{w}||}$, таким чином задача зводиться до мінімізації $\left|\left|\mathrm{w}\right|\right|$.

З метою виключення усіх точок зі смуги, треба впевнитись, що ∀i:

$$\begin{bmatrix} w \cdot x_i - b \geq 1, c_i = 1 \\ w \cdot x_i - b \leq -1, c_i = -1 \end{bmatrix} \quad (2.4)$$

(2.4) також може бути записано у вигляді (2.5).

$$c_i(x \cdot x_i - b) \geq 1, 1 \leq i \leq n \text{ (11)} \quad (2.5)$$

[31] Тож, у випадку лінійної роздільності даних, задача побудови оптимальної розділюючої гіперплощини зводиться до мінімізації $||w||$ за умови (2.4). Це задача квадратичної оптимізації, що має вигляд:

$$\begin{cases} ||w||^2 \to min \\ c_i(w \cdot x_i - b) \geq 1 \, , 1 \leq i \leq n \end{cases}$$

За теоремою Куна-Такера ця задача еквівалентна двоїстій задачі пошуку сідлової точки функції Лагранжа:

$$\begin{cases} L(w, b; \lambda) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \lambda_i(c_i((w \cdot x_i) - b) - 1) \to min_{w,b} \max_{\lambda}) \\ \lambda_i \geq 0, \;\; 1 \leq i \leq n \end{cases} \quad (2.6)$$

тут $\lambda = (\lambda_1 \ldots \lambda_n)$ −вектор двоїстих змінних.

Задача (2.6) зводиться до еквівалентної задачі квадратичного програмування, що містить тільки двоїсті змінні:

$$\begin{cases} -L(\lambda) = -\sum_{i=1}^{n} \lambda_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j c_i c_j (x_i \cdot x_j \to \min_{\lambda}) \\ \lambda_i \geq 0, \;\; 1 \leq i \leq n \\ \sum_{i=1}^{n} \lambda_i c_i = 0 \end{cases}$$

Розв’язавши її, можна знайти $w = \sum_{i=1}^{n} \lambda_i c_i x_i$, $b = w \cdot x_i - c_i, \lambda_i > 0$

В результаті, алгоритм класифікації можна записати наступним чином:

$$a(x) = sign(\sum_{i=1}^{n} \lambda_i c_i x_i - b) \quad (2.7)$$

При цьому сумування йде не по усій вибірці даних, а тільки по опорним векторам, для яких $\lambda_i \neq 0$.

Вище був розглянутий випадок лінійної роздільності, проте на практиці частіше мають справу з лінійно нероздільними даними. В такому випадку вводяться додаткові змінні $\xi_i \geq 0$, що характеризують величину помилки на прикладах x_i, $1 \leq i \leq n$. Продовжимо з (2.6), пом'якшемо обмеження нерівності, а також введемо штраф за сумарну помилку:

$$\begin{cases} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \to \min_{w,b,\xi_i} \\ c_i(w \cdot x_i - b) \geq 1 - \xi_i, \ \ 1 \leq i \leq n \\ \xi_i \geq 0, \ \ 1 \leq i \leq n \end{cases}$$

Коефіцієнт C −параметр налаштування методу, що дозволяє регулювати відношення між максимізацією ширини розділюючої смуги та мінімізацією сумарної помилки.

Аналогічно, за теоремою Куна-Такера можна звести задачу до пошуку сідлової точки функції Лагранжа:

$$\begin{cases} L(w,b,\xi;\lambda,\eta) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \lambda_i\big(c_i\big((w \cdot x_i) - b\big) - 1\big) - \\ \quad - \sum_{i=1}^{n} \xi_i(\lambda_i + \eta_i - C) \to \min_{w,b,\xi}\max_{\lambda,\eta} \\ \xi_i \geq 0, \lambda_i \geq 0, \eta_i \geq 0, \quad 1 \leq i \leq n \\ \begin{bmatrix} \lambda_i = 0 \\ c_i\big((w \cdot x_i) - b\big) = 1 - \xi_i \end{bmatrix}, 1 \leq i \leq n \\ \begin{bmatrix} \eta_i = 0 \\ \xi_i = 0 \end{bmatrix}, 1 \leq i \leq n \end{cases}$$

Аналогічно зведемо цю задачу до еквівалентної:

$$\begin{cases} -L(\lambda) = -\sum_{i=1}^{n} \lambda_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j c_i c_j (x_i \cdot x_j \to \min_{\lambda}) \\ 0 \leq \lambda_i \geq C, \ \ 1 \leq i \leq n \\ \sum_{i=1}^{n} \lambda_i c_i = 0 \end{cases}$$

Для алгоритму класифікації зберігається формула (2.7), тільки тепер ненульові значення λ_i мають не тільки опорні об'єкти, але і «об'єкти-

порушники». В певному сенсі це є недоліком, оскільки «порушниками» часто виявляються шумові викиди, а побудоване на них вирішальне правило, фактично, спирається на шум.

2.7 Градієнтний бустинг

Коли ми намагаємося передбачити цільову змінну, а саме відтік, за допомогою будь-якого одиничного алгоритму машинного навчання, відмінності між реальною міткою класу та передбаченою моделлю можуть бути викликані такими чинниками як шум (noise), розбіжність (variance) і зміщення (bias). А використання ансамблей (поєднання декількох навчених алгоритмів з метою отримання кращої прогностичної ефективності) допомагає зменшити вплив цих факторів. Поширеними ансамблевими методами, що підвищують точність прогнозування, є беггінг та бустинг. Основна відмінність між ними - метод навчання. У випадку з беггінгом фахівці підвищують точність слабких моделей, паралельно навчаючи деякі з них на різних наборах даних. Бустинг же навчає слабкі моделі послідовно.

Градієнтний бустинг - це техніка машинного навчання, яка будує модель передбачення у вигляді ансамблю слабких моделей, якими зазвичай є дерева рішень. З кількох слабких моделей у підсумку ми збираємо одну, але вже ефективну. На кожній ітерації обчислюються відхилення прогнозів уже навченого ансамблю на навчальній вибірці. Наступна модель, яку буде додано в ансамбль, передбачатиме ці відхилення. Таким чином, додавши передбачення нового дерева до передбачень навченого ансамблю, ми можемо зменшити середнє відхилення моделі, яке є таргетом оптимізаційної задачі. Нові дерева додаються до ансамблю доти, доки помилка зменшується, або доки не виконується одне з правил "ранньої зупинки". Надалі для програмної реалізації даного метода машинного навчання використовуватиметься

бібліотека XGBoost — одна з найпопулярніших та найефективніших реалізацій алгоритму градієнтного бустингу на деревах рішень.

До переваг градієнтного бустингу з використанням бібліотеки XGBoost належить.

1. Висока швидкість: XGBoost використовує оптимізовані алгоритми та структуру даних, що забезпечує відносно високу швидкість розрахунків. Крім того, XGBoost може використовувати паралельну обробку на багатьох ядрах процесора, що також дозволяє прискорити навчання моделі.

2. Обробка пропущених значень: XGBoost має вбудовану обробку пропущених значень, що дозволяє автоматично обробляти відсутні дані без необхідності виконувати попередню обробку даних.

3. Висока точність: XGBoost використовує різні техніки оптимізації, такі як наприклад регуляризація, що дозволяє покращити точність моделі і зменшити ризик її перенавчання.

За алгоритмом лежить наступна математика[32]:

$\Lambda^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$ − функція для оптимізації градієнтного бустингу, де:

l −функція втрат

$y_i, \hat{y}_i^{(t)}$ −значення залежної змінної і-го прикладу навчальної вибірки даних та сума передбачень перших t дерев відповідно.

x_i − набір характеристик і-го прикладу навчальної вибірки даних

f_t − функція(наразі дерево рішень), яка буде навчатися на кроці t.

$f_t(x_i)$ − передбачення для і-го прикладу навчальної вибірки

$\Omega(f_t)$ − регуляризація функції f. $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda||w||$, де T −кількість вершин в дереві, w −значення в листях, а γ та λ −параметри регуляризації.

Далі за допомогою розкладу в ряд Тейлора до другого члена можемо наблизити функцію $\Lambda^{(t)}$ наступним чином:

$$\Lambda^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + g_i f_t(x_i) + 0.5 h_i f_t^2(x_i)\right) + \Omega(f_t), \qquad (2.12)$$

де $g_i = \frac{\partial l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial \hat{y}_i^{(t-1)}}, h_i = \frac{\partial^2 l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial^2 \hat{y}_i^{(t-1)}}$

Оскільки нашою метою є мінімізувати помилку моделі на навчальній вибірці, нам необхідно знайти мінімум $\Lambda^{(t)}$ для кожного t. Мінімум виразу (2.12) відносно $f_t(x_i)$ знаходиться в точці $f_t(x_i) = -\frac{g_i}{h_i}$.

Кожне окреме дерево $f_t(x_i)$ навчається за стандартним алгоритмом.

2.8 Оцінювання якості моделей

Попередня обробка даних, побудова моделей та налаштування оптимальних параметрів є важливими етапами в розв’язанні поставленої задачі класифікації, проте не менш важливою частиною процесу машинного навчання є перевірка якості побудованих моделей, оскільки дозволяє визначити найефективнішу, ту, яка здатна забезпечити найвищу точність прогнозу та може бути успішно використана на практиці.

Для досягнення найкращих результатів і забезпечення надійних прогнозів, дослідники та практики широко використовують різні методи та метрики оцінювання моделей машинного навчання.

Навчання та тестування моделі на одних і тих же даних не є ефективним, оскільки це не дає об'єктивної оцінки її реальної продуктивності. Модель просто запам'ятовує вхідні приклади і мітки, а потім відтворює їх, що призводить до переоцінки точності та надмірної впевненості в її результатах.

Розбивка даних на набори для навчання і тестування має вирішальне значення для зниження цього ризику. Однак досить складно розділити набір даних так, щоб максимізувати ефективність навчання і достовірність результатів тестування. В ході виконання цієї роботи буде використовуватися перехресна перевірка як ефективний інструмент для вирішення описаної проблеми. Цей статистичний метод дозволяє оцінити модель на незалежних

даних, які не використовуються для навчання. Кожен цикл перехресної перевірки включає випадкове розбиття вихідного набору даних на навчальний і тестовий набори. Навчальний набір використовується для навчання алгоритму, а тестовий набір - для оцінювання її якості, цей процес повторюється кілька разів, а як показник ефективності навчання використовується середня помилка крос-валідації.

Існує декілька методів перехресної перевірки, але надалі застосовуватиметься k-кратна крос-валідація (k-fold), що розбиває дані на k випадково обраних наборів приблизно однакового розміру. Один набір використовуватиметься для тестування моделі, навченої на інших наборах даних. Цей процес буде повторюватись k разів, так що кожен набір використовуватиметься для перевірки рівно один раз. Це один із найпопулярніших методів перехресної перевірки, але його виконання зазвичай займає багато часу, оскільки модель потрібно навчати багаторазово. На рисунку 2.5 зображено процес k-кратної перехресної перевірки[33]:

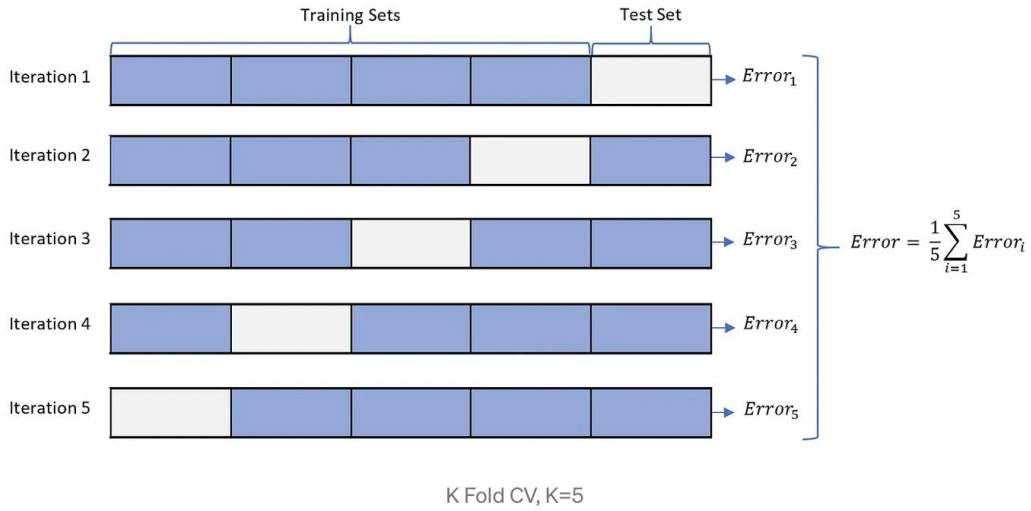


Рисунок 2.5 – k-кратна перехресна перевірка

Тож, перехресна перевірка є простою для розуміння та реалізації і забезпечує меншу упередженість в оцінці якості порівняно з іншими методами.

До всього вищезгаданого, для оцінки якості моделей використовуються різні метрики, в даній роботі буде побудовано матрицю помилок (confusion matrix), а також розглянуто такі метрики як: $accuracy, precision, recall.$

Матриця помилок – це таблиця, яка дозволяє візуалізувати ефективність алгоритму класифікації шляхом порівняння прогнозованого значення цільової змінної з її фактичним значенням [34]. На рисунку 2.6 зображено вигляд матриці помилок:

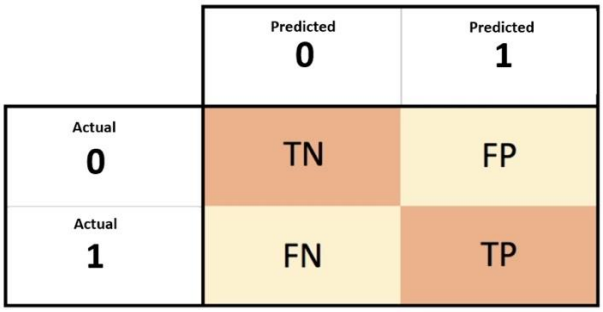


Рисунок 2.6 – матриця помилок (confusion matrix)

Тут, TP — true-positive, FP — false-positive, TN — true-negative, FN — false-negative

Будь який реальний класифікатор робить помилки. На рисунку 2.6 можна побачити матрицю помилок для бінарної класифікації. В умовах поставленої задачі, побудована модель по певному набору параметрів прикладу має віднести його до одного з двух класів (0 – категорія людей, що не є на межі відтоку, 1 - користувачі, що на межі припинення взаємодії з компанією). Predicted 0/1 – прогнозовані моделлю мітки прикладів, Actual 0/1 – реальні мітки. Тож, під час присвоєння моделлю мітки прикладу виникають помилки першого та другого роду. Помилка I роду, або хибно-позитивний (*false-positive)* результат класифікації, має місце, коли негативне спостереження розпізнане моделлю як позитивне. Помилкою II роду, або хибно-негативним (*false-negative*) результатом класифікації, називають випадок, коли позитивне спостереження розпізнано як негативне. Певно, що в межах поставленої задачі важливішою є помилка другого роду, адже в такому випадку алгоритм пропускає людину, що знаходиться на межі відтоку,

класифікуючи її як таку, що не має намірів піти від компанії, а це не відповідає реальності, і відповідно постачальник послуг не може вчасно вжити заходів для підвищення її лояльності до компанії, як наслідок втрачає абонента.

На основі матриці помилок та її значень, розраховуються різні метрики класифікаційної здатності алгоритму, тож в межах оцінки побудованих моделей розглядатимуться також наступні метрики [35].

1. Accuracy - широко використовувана і легка для розуміння метрика. Це відношення всіх правильних прогнозів до загальної кількості всіх передбачень. Фактично це ймовірність того, що клас прикладу буде передбачено правильно.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Precision - це частка правильних відповідей моделі в межах класу - це частка об'єктів, що справді належать даному класу відносно всіх об'єктів, які система віднесла до цього класу.

$$Precision = \frac{TP}{TP + FP}$$

Саме перевірка precision не дає нам змоги записувати всі об'єкти в один клас, оскільки в цьому разі ми отримуємо зростання рівня False Positive.

3. Recall - це частка істинно позитивних класифікацій. Повнота показує, яку частку об'єктів, що реально належать до позитивного класу, модель передбачила правильно. Інакше кажучи, повнота характеризує втрати класифікатора на оцінювальній вибірці - чим вище це значення, тим менше правильних класифікацій було пропущено. Для задачі прогнозування відтоку ця метрика буде особливо важливою, адже вона показує наскільки добре модель розпізнає людей, що є на межі відтоку.

$$Recall = \frac{TP}{TP + FN}$$

4. Precision і recall не залежать, на відміну від accuracy, від співвідношення класів і тому є інформативними в умовах незбалансованих вибірок. В реальності часто на практиці стоїть завдання знайти баланс між

цими двома метриками. Зрозуміло, що чим вища точність і повнота, тим краще. Але в реальному житті максимальна точність і повнота недосяжні одночасно і доводиться шукати оптимальне рішення. Тому в роботі перевіряється також метрика F-міра , яка об'єднує у собі інформацію про точність і повноту моделі.

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall}$$

Тут $\beta - \text{ваги, що}$ приймає значення у діапазоні (0,1) якщо при розрахунку F-міри дослідник хоче надати перевагу точності, значення >1, повноті та =1, якщо внесок точності і повноти у розрахунок F-міри вважається однаковим.

І на останок, варто згадати про не менш інформативний інструмент для оцінки якості моделі - ROC-криву. Це графічне відображення залежності між True Positive Rate (TPR) і False Positive Rate (FPR) для різних порогових значень класифікаційної моделі. TPR відображає частку правильно класифікованих об'єктів позитивного класу, а FPR відображає частку неправильно класифікованих об'єктів негативного класу. ROC-крива дозволяє зрозуміти, як добре модель розрізняє класи, а не просто дивитися на точність передбачень при конкретному пороговому значенні. А AUC-ROC - це площа під ROC-кривою. AUC-ROC вимірює загальну здатність моделі розпізнавати класи та розподіляти їх відповідно. Значення AUC-ROC знаходиться в діапазоні від 0 до 1, де значення 1 вказує на ідеальну модель, яка розділяє класи безпомилково, а значення нижче 0.5 вказує на модель, яка працює гірше, ніж випадкова класифікація.

2.9 Висновки до розділу 2

В даному розділі було розглянуто машинне навчання як ефективний інструмент для вирішення задач класифікації. А саме проаналізовано окремі методи машинного навчання, які використовуватимуться для розв’язання поставленої в першому розділі задачі. Кожен з цих алгоритмів має свої переваги та недоліки, а їх точність залежить від багатьох факторів. Продовженням дослідження буде перевірка роботи цих алгоритмів на практиці.

В цілому, враховуючи ефективність машинного навчання та розглянуті алгоритми, ми можемо очікувати успішні результати у вирішенні поставленої задачі класифікації. Однак, необхідно провести подальшу роботу та аналіз, щоб з'ясувати, як точність цих алгоритмів залежить від конкретних даних та факторів.

3 ПРОЦЕС МОДЕЛЮВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ ПРОГНОЗУВАННЯ

3.1 Вступ до розділу 3

Прогнозування відтоку користувачів в телекомунікаційній галузі за допомогою методів машинного навчання має на меті побудову такої інтелектуальної системи, що зможе допомогти компаніям з досить високою точністю виявляти користувачів, що хочуть відмовитись від послуг оператора, а також розуміти основні фактори, що впливають на цей процес. Однак для досягнення цієї мети мало просто використати описані в попередньому розділі моделі машинного навчання, адже їх якість і прогностична сила залежить від багатьох факторів, які необхідно врахувати ще до моделювання.

Повний цикл прогнозування відтоку складається з кількох етапів, кожен з яких наближує нас до мети і так чи інакше впливає на загальний результат.

1. Збір даних, необхідних для проведення будь якого аналізу. Цей етап включає збір інформації про клієнтів, їхні характеристики, історію користування послугами тощо. Компанії в телекомунікаційній галузі зазвичай мають великі бази даних і доступ до різних джерел, що дозволяє їм прослідковувати важливі процеси та збирати цінну інформацію щодня.

2. Аналіз та візуалізація даних. Цей етап включає дослідження зібраних даних, що дозволяє виявити кореляції, тренди та патерни, які можуть допомогти зрозуміти поведінку клієнтів. Візуалізація забезпечує зрозуміле і наочне представлення даних, що полегшує їх аналіз.

3. Попередня обробка. Зібрані дані часто мають незадовільну якість для подальшого моделювання, вони можуть містити шуми, пропущені чи помилкові значення, неузгоджені типи даних, все це може потворювати

прогнозування, тож важливим етапом є їх попередня обробка перш ніж навчати на них моделі.

4. Моделювання. Вже на цьому етапі можна переходити до побудови прогностичних моделей машинного навчання.

5. Оцінювання якості моделей. Останній етап включає оцінку якості побудованих моделей прогнозування. Оцінювання дозволяє визначити ефективність моделей і внести необхідні корективи для покращення їх результатів.

3.2 Вибір мови програмування та опис бібліотек

Подальша робота проводитиметься з допомогою мови Python. Загалом, Python є потужним, зручним у використанні та широко підтримуваним інструментом, що часто застосовується для реалізації проектів з машинного навчання. Мова пропонує багату екосистему бібліотек та фреймворків, таких як NumPy, Pandas, Matplotlib, TensorFlow, scikit-learn, що надають потужні інструменти для реалізації алгоритмів прогнозування, обробки даних, візуалізації тощо. А також Python легко поєднується з іншими інструментами та мовами програмування, що дає можливість інтегрувати розроблені моделі машинного навчання у вже існуючі системи, бази даних, веб-сервери та інші компоненти інфраструктури.

У роботі були використані наступні бібліотеки [36]:

1) numpy та pandas – дві основні бібліотеки для обробки та аналізу даних;
2) imbalanced-learn – бібліотекою Python, яка надає набір інструментів для роботи з незбалансованими наборами даних. З даної бібліотеки в роботі були використані такі методи як SMOTE та RandomUnderSampler;
3) matplotlib – це комплексна бібліотека для створення статичних, та інтерактивних візуалізацій, використовується у роботі для побудови графіків;

4) seaborn – бібліотека для створення статистичних графіків на Python. Вона побудована на основі matplotlib і тісно інтегрується зі структурами даних pandas;

5) sklearn.preprocessing – модуль бібліотеки scikit-learn, що надає різноманітні інструменти для попередньої обробки даних. Зокрема в роботі використовуються функції LabelEncoder, OneHotEncoder, MinMaxScaler;

6) sklearn.model_selection – модуль бібліотеки scikit-learn, який містить інструменти для вибору та оцінки моделей машинного навчання. В роботі були використані такі фунції та класи як train_test_split, GridSearchCV, cross_val_score, Kfold;

7) sklearn.tree – модуль бібліотеки scikit-learn, який містить імплементації алгоритмів дерев рішень для завдань класифікації та регресії;

8) XGBoost – високоефективна бібліотека для градієнтного бустингу дерев рішень;

9) sklearn.metrics – модуль бібліотеки scikit-learn, який надає різноманітні метрики для оцінки якості моделей машинного навчання;

10) sklearn.svm – модуль бібліотеки scikit-learn, який містить різні реалізації методів опорних векторів (Support Vector Machines, SVM) для задач класифікації і регресії;

11) sklearn.naive_bayes – модуль бібліотеки scikit-learn, який містить реалізації алгоритмів наївного баєсівського класифікатора;

12) sklearn.linear_model – модуль бібліотеки scikit-learn, який містить реалізації лінійних моделей, зокрема в роботі використовується логістична регресія.

3.3 Описовий аналіз вхідних даних

Для проведення дослідження в даній роботі використовуватиметься набір даних представлений IBM [37], що є в загальному доступі і містить інформацію про клієнтів телекомунікаційної компанії (клієнтські дані, інформацію про послуги, якими вони користуються, історію взаємодії з компанією тощо). Враховуючи різноманітність даних у цьому датасеті, він є корисним ресурсом для проведення подібних досліджень.

Тож перш за все, імпортуємо усі необхідні для подальшої роботи бібліотеки та дані, на основі яких проводитиметься подальше моделювання. Цей процес зображений на рисунку 3.1.

```
link = 'https://drive.google.com/file/d/1tbgtf2us4bmZXWUDk9_EpgevwA1sI3Ri/view'
id = link.split("/")[-2]
downloaded = drive.CreateFile({'id':id})
downloaded.GetContentFile('telecom_customer_churn_final.csv')
telecom_customers = pd.read_csv('telecom_customer_churn_final.csv')
```

Рисунок 3.1 – Імпортування навчального набору даних

Використаємо відповідні методи об’єктів бібліотеки pundas для опису даних. Подивимось на їх розмір та детальний опис змінних на рисунку 3.2.

```
telecom_customers.shape
```

```
(7043, 23)
```

```
# Let's explore variables, their data types, and total non-null values
telecom_customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   ZIPcode           7043 non-null   int64  
 4   Partner           7043 non-null   object 
 5   Dependents        7043 non-null   object 
 6   tenure            7043 non-null   int64  
 7   PhoneService      7043 non-null   object 
 8   MultipleLines     7043 non-null   object 
 9   InternetService   7043 non-null   object 
 10  OnlineSecurity    7043 non-null   object 
 11  OnlineBackup      7043 non-null   object 
 12  DeviceProtection  7043 non-null   object 
 13  TechSupport       7043 non-null   object 
 14  StreamingTV       7043 non-null   object 
 15  StreamingMovies   7043 non-null   object 
 16  Contract          7043 non-null   object 
 17  PaperlessBilling  7043 non-null   object 
 18  PaymentMethod     7043 non-null   object 
 19  MonthlyCharges    7043 non-null   float64
 20  TotalCharges      7043 non-null   object 
 21  Churn             7043 non-null   object 
 22  ChurnReason       1869 non-null   object 
dtypes: float64(1), int64(3), object(19)
```

Рисунок 3.2 – Структура та зміст даних навчальної вибірки

Переглянемо також п’ять перших об’єктів навчального набору даних на рисунку 3.3.

```
# Check the 5 first rows of the dataset.
telecom_customers.head()
```

	customerID	gender	SeniorCitizen	ZIPcode	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	TechSupport
0	7590-VHVEG	Female	0	90003	Yes	No	1	No	No phone service	DSL	...	No
1	5575-GNVDE	Male	0	90005	No	No	34	Yes	No	DSL	...	No
2	3668-QPYBK	Male	0	90006	No	No	2	Yes	No	DSL	...	No
3	7795-CFOCW	Male	0	90010	No	No	45	No	No phone service	DSL	...	Yes
4	9237-HQITU	Female	0	90015	No	No	2	Yes	No	Fiber optic	...	No

5 rows × 23 columns

Рисунок 3.3 – Початковий вигляд даних

Після первинного огляду навального набору даних можна зробити висновки, що датасет містить інформацію про 7043 абонентів, кожен з яких характеризується 23-ма характеристиками, на основі яких в подальшому прийматиметься рішення щодо класифікації. Цільовою змінною є атрибут Churn.

Усі вищезазначені характеристики умовно можна розділити на три групи, а саме.

1) Демографічні характеристики:
   - Gender - стать абонента;
   - SeniorCitizen - вікова категорія абонента, в даному випадку чи є він людиною похилого віку;
   - Partner - характеристика, що вказує на наявність у абонента партнера;
   - Dependents - характеристика, що вказує на наявність у абонента утриманців;
   - ZIPcode - поштовий індекс абонента;
2) Інформація про стан рахунку користувача:

– Tenure - кількість місяців протягом який абонент взаємодіє з компанією;

– Contract - вказує на поточний тип контракту клієнта (помісячний, річний, на два роки);

– PaperlessBilling - характеристика, що вказує чи користується клієнт сервісом безпаперового білінгу;

– PaymentMethod - спосіб оплати за послуги;

– MontlyCharges - сума, що стягається з клієнта щомісяця;

– TotalCharges - загальна сума, стягнута з клієнта за період користування послугами оператора.

3) Інформація про сервіси, використовувані абонентами:

Усі характеристики нижче не вимагають пояснення, адже означають наявність(відсутність) підписки на дану послугу абонентом.

– PhoneService

– MultipleLines

– InternetServices

– OnlineSecurity

– OnlineBackup

– DeviceProtection

– TechSupport

– StreamingTV:

– StreamingMovies

Окремо варто вказати про такі характеристики як customerID та ChurnReason. ChurnReason – атрибут, що містить інформацію про причину розірвання контракту абонентами, що покинули компанію, ця характеристика буде використана надалі для аналізу причин відтоку аудиторії. customerID є унікальним ідентифікатором користувача послуг і не несе в собі інформації, корисної для аналізу, тож варто відразу видалити цей атрибут. Також з рисунку 3.2 бачимо, що TotalCharges має тим даних object, хоча за своїм змістом має бути числовою змінною і характеризувати витрати абонентів, тож

відразу переведемо її у відповідний тип даних. Вигляд оновленого датасету на рисунку 3.4.

```
# Converting Total Charges to a numerical data type.
telecom_customers.TotalCharges = pd.to_numeric(telecom_customers.TotalCharges, errors='coerce')
```

```
telecom_customers.drop(columns='customerID', inplace=True)
telecom_customers
```

	gender	SeniorCitizen	ZIPcode	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	Female	0	90003	Yes	No	1	No	No phone service	DSL	No	...
1	Male	0	90005	No	No	34	Yes	No	DSL	Yes	...
2	Male	0	90006	No	No	2	Yes	No	DSL	Yes	...

Рисунок 3.4 – Вигляд даних після оновлення

Продовжимо аналіз даних та подивимось на опис числових характеристик, а також на кореляцію між ними на рисунку 3.5.

```
# Find correlations
telecom_customers.corr(method='pearson')
```

	SeniorCitizen	ZIPcode	tenure	MonthlyCharges
SeniorCitizen	1.000000	0.004968	0.016567	0.220173
ZIPcode	0.004968	1.000000	0.022422	0.006300
tenure	0.016567	0.022422	1.000000	0.247900
MonthlyCharges	0.220173	0.006300	0.247900	1.000000

```
# Describe columns with numerical values
telecom_customers.describe()
```

	SeniorCitizen	ZIPcode	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000	7032.000000
mean	0.162147	93521.964646	32.371149	64.761692	2283.300441
std	0.368612	1865.794555	24.559481	30.090047	2266.771362
min	0.000000	90001.000000	0.000000	18.250000	18.800000
25%	0.000000	92102.000000	9.000000	35.500000	401.450000
50%	0.000000	93552.000000	29.000000	70.350000	1397.475000
75%	0.000000	95351.000000	55.000000	89.850000	3794.737500
max	1.000000	96161.000000	72.000000	118.750000	8684.800000

Рисунок 3.5 – Опис числових характеристик

Виведемо також категоріальні змінні, подивимось на значення, яких вони набувають та чисельне співвідношення між ними на рисунку 3.6.

```
categorial_features = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod']
number_features = ['tenure','MonthlyCharges', 'TotalCharges']

for i in cat_features: #Checking Data
    print(i)
    print(churn[i].value_counts())
    print("----------------------------------------------------")
```

```
gender
Female    939
Male      930
Name: gender, dtype: int64
----------------------------------------------------
SeniorCitizen
0    1393
1     476
Name: SeniorCitizen, dtype: int64
----------------------------------------------------
Partner
No     1200
Yes     669
Name: Partner, dtype: int64
----------------------------------------------------
Dependents
No     1543
Yes     326
Name: Dependents, dtype: int64
----------------------------------------------------
PhoneService
Yes    1699
No      170
Name: PhoneService, dtype: int64
----------------------------------------------------
```

Рисунок 3.6 – Опис категоріальних змінних

3.4 Візуалізація вхідних даних

Ми вже маємо певне уявлення про наповнення датасету, проте для його кращого розуміння необхідно провести візуалізацію наданих нам даних (рис. 3.7).

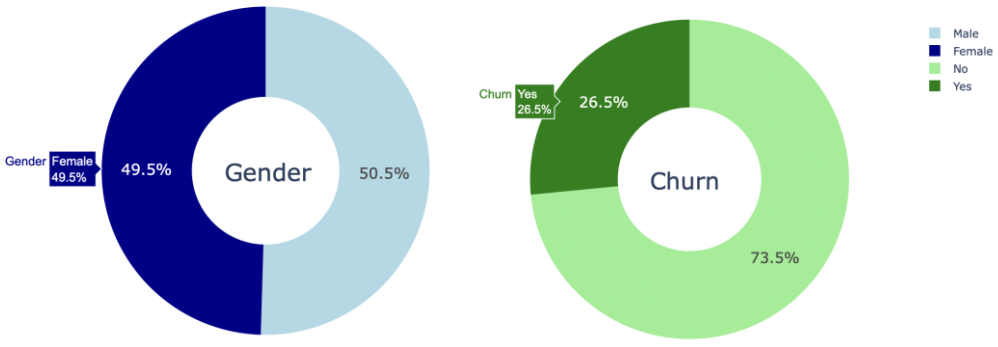


Рисунок 3.7 – Співвідношення досліджуваних класів

Як бачимо з кругової діаграми на рисунку 3.7 дані є незбалансованими, це вказує на нерівномірний розподіл класів (у даному випадку, користувачів, які відтікають і ті що не мають таких намірів). 73.5% аудиторії компанії продовжує активну взаємодію з нею і тільки 26.5% має наміри розірвати контракт. Для бізнесу, який має такі показники, це є непогана ситуація, проте тренування моделей передбачення на таких даних може призвести до нерелевантних результатів. Дисбаланс класів часто спричиняє спотворення результатів моделі та заниження її реальної ефективності. Якщо в подальшому дослідженні використовувати ці дані без врахування вищезазначеного фактору, моделі можуть бути схильними до передбачення більш численного класу, нехтуючи менш чисельним, що у випадку прогнозування відтоку є недопустимим, адже для компанії особливо важливо розуміти ту категорію користувачів, що з певних причин хоче відмовитись від послуг, які надає оператор.

Датасет, на основі якого проводиться дослідження, містить таку ознаку як ChurnReason, саме вона і допоможе проаналізувати причини, які спонукають споживачів шукати нових постачальників телекомунікаційних послуг. Відповідний графік зображений на рисунку 3.8.

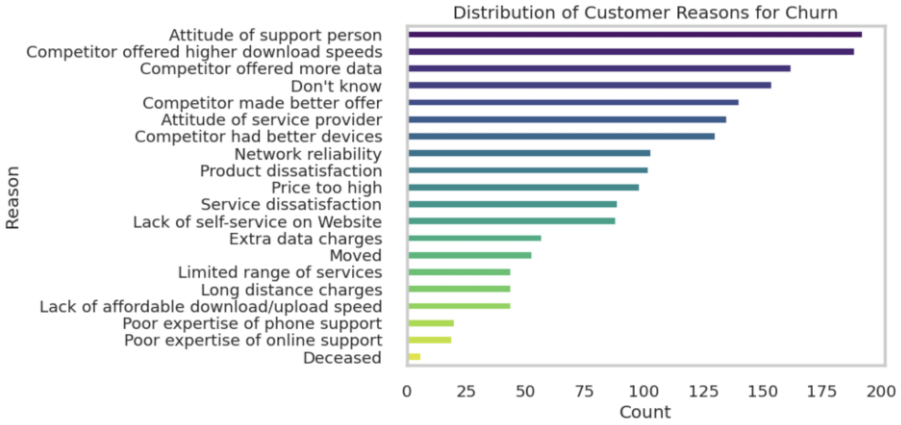


Рисунок 3.8 – Причини відтоку користувачів

На рисунку 3.8 відображено 20 причин чому користувачі роблять вибір на користь іншої компанії. Як бачимо основними є якість обслуговування та

сервісу, який надає оператор, і власне той фактор, що конкуренти мають цікавіші пропозиції (більше даних, стабільнішу мережу, швидший обмін даними тощо).

Проаналізуємо демографічні характеристики користувачів на рисунку 3.9. Кожен графік показує розподіл кількості спостережень для кожної категорії демографічних ознак залежно від значення цільової змінної Churn.

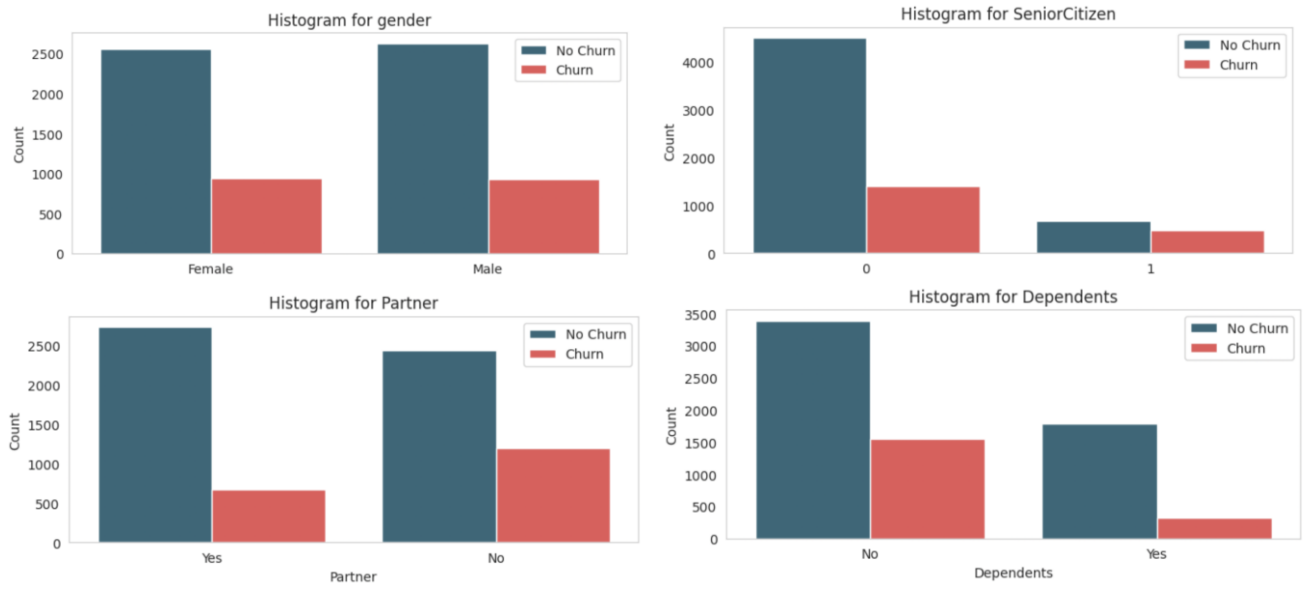


Рисунок 3.9 – Демографічні характеристики користувачів

Можемо зробити наступні висновки: відтік серед людей старшого віку майже вдвічі вищий, ніж серед молоді; ми не очікуємо, що стать має значну прогностичну силу, це й підтверджується графіком, відсоток відтоку не залежить від того, чи є клієнт чоловіком або жінкою; клієнти, які мають партнера, відмовляються від послуг компанії частіше, ніж ті, що такого не мають.

Також подивимось на інформацію про облікові записи користувачів на рисунку 3.10, що є важливим для компанії, оскільки ці характеристики датасету містить дані про платежі та рахунки клієнтів.

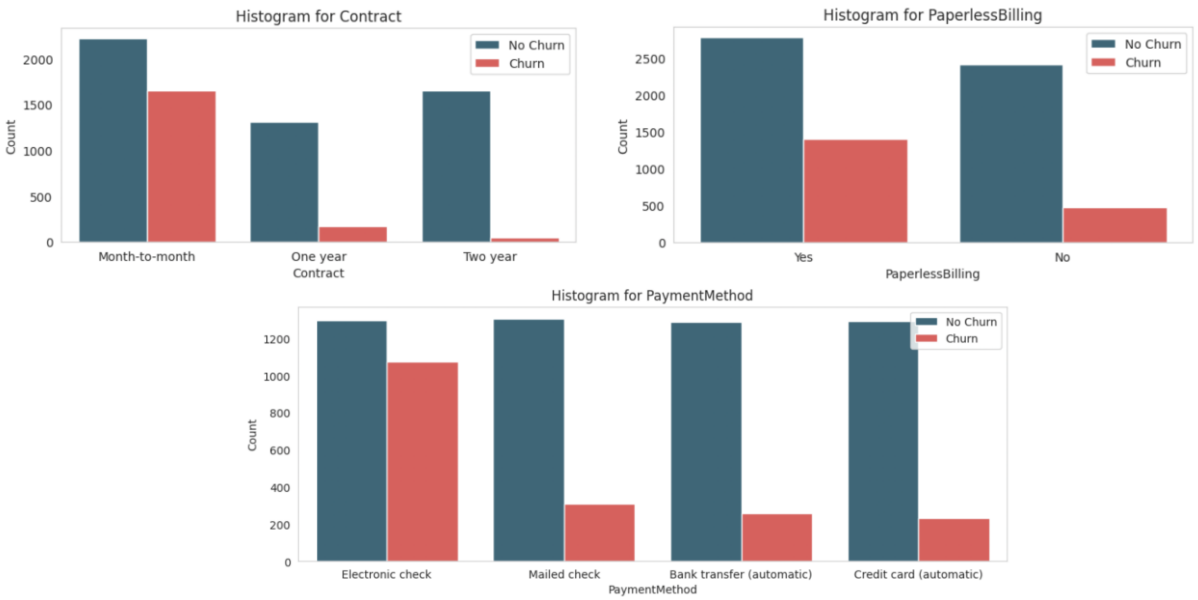


Рисунок 3.10 – Характеристики облікових записів

Бачимо, що користувачі, які мають контракти на місць, підписалися безпаперовий білінг та обрали електронний чек як спосіб оплати мають найвищий рівень відтоку. Це може бути зумовлено тим, що такі абоненти більш гнучкі, їм легше змінити постачальника послуг, а відповідно і прийняти таке рішення вони можуть доволі швидко. Основуючись на цій інформації оператор може приймати релевантні стратегічні рішення для покращення користувацького досвіду з метою зменшення відтоку, наприклад спробувати внести зміни в умови контрактів термінами на рік чи два, зробити їх привабливішими.

Вище були розглянуті категоріальні змінні, тепер подивимось на чисельні ознаки. Графік boxplot на рисунку 3.11 відображає розподіл значень числової змінної, а також надає інформацію про медіану і викиди в даних.

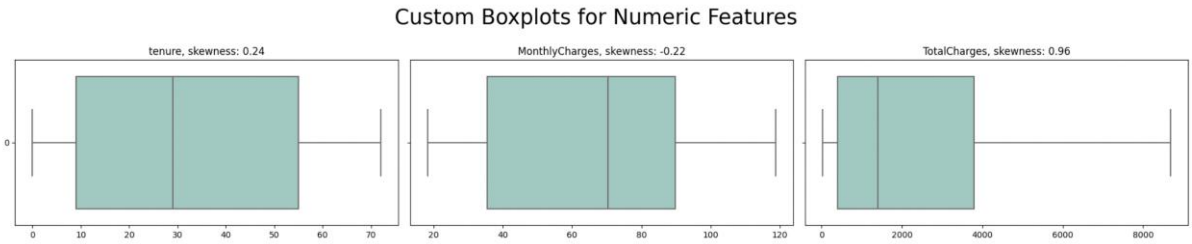


Рисунок 3.11 – Boxplots для чисельних ознак

Як бачимо, викидів серед числових змінних немає. Tenure та MonthlyCharges мають бімодальний розподіл, а TotalCharges має правосторонню асиметрію (right skew). Це означає, що більша частина значень знаходиться в лівій частині розподілу, а менша - в правій частині. Тобто користувачів, у яких загальні витрати менші, - більше.

Розглянемо ці ж атрибути залежно від цільової змінної на рисунку 3.12.

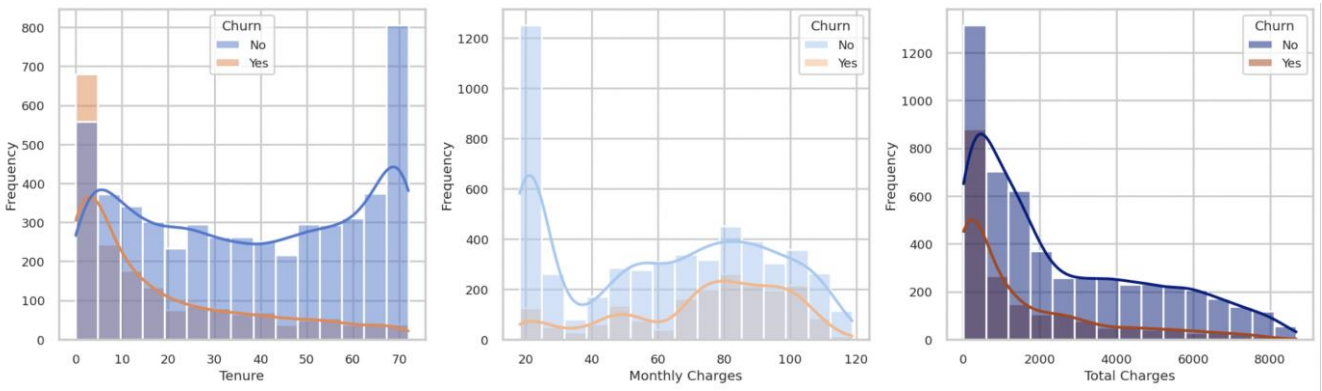


Рисунок 3.12 – Розподіл чисельних ознак відносно цільової змінної

Спостерігаємо збільшення відтоку користувачів зі збільшенням місячної плати за послуги. Клієнти можуть відчувати, що отримують менше цінності від сервісу, що надається оператором, порівняно з сумою, яку вони платять, що спонукає їх до пошуку вигідніших пропозицій. Також бачимо, що нові користувачі більш схильні до відтоку, це може бути пов’язано з тим фактом, що компанія не встигла в короткі терміни завоювати лояльність абонента. До того ж, варто зауважити, що користувачі з високими загальними витратами рідше залишають компанію. Як правило, високі рахунки свідчать про довіру аудиторії та про стійкі взаємовідносини з оператором.

Залишилось подивитись на дані, що містять інформацію про користування сервісами, що пропонує оператор на рисунку 3.13.

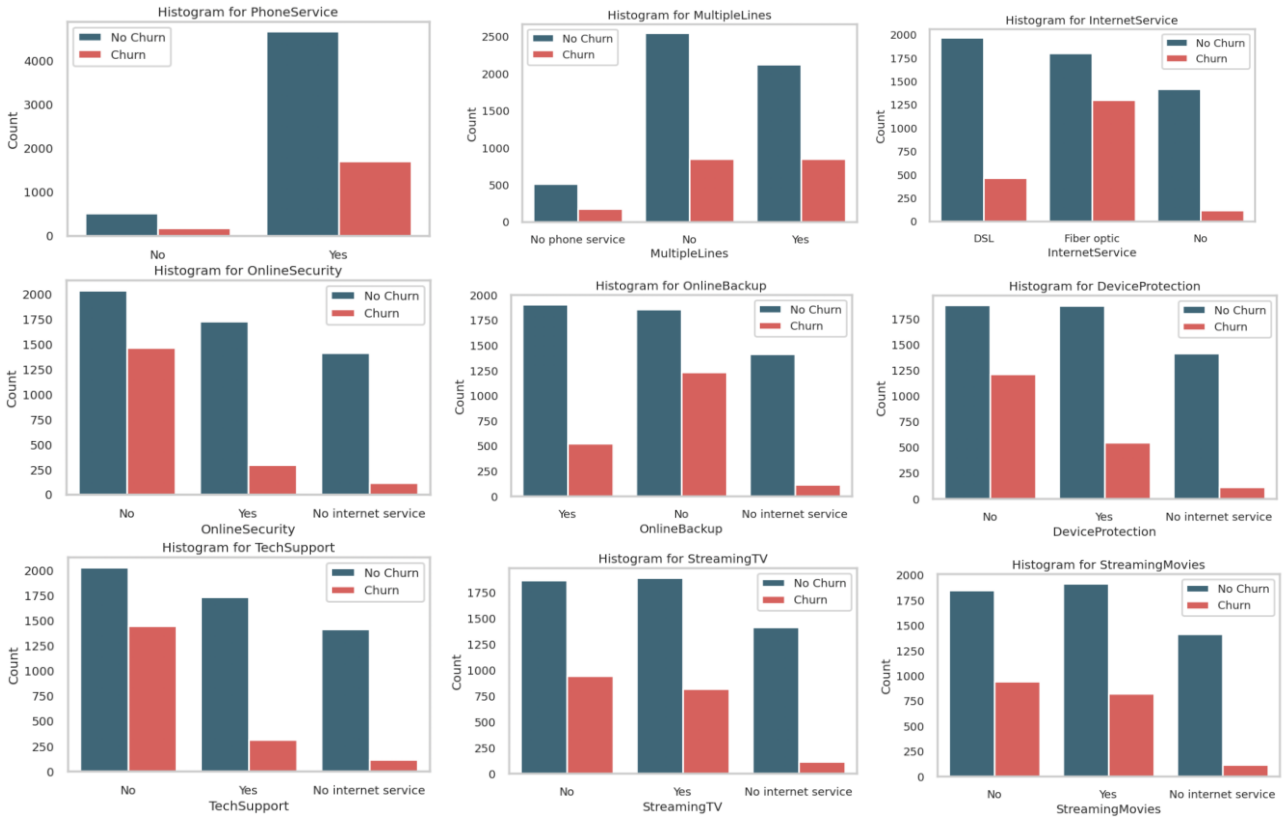


Рисунок 3.13 – Інформація про користування сервісами

З рисунку 3.13 можна зробити деякі висновки, наприклад, MultipleLines не мають значної прогностичної сили, відсоток відтоку для класів майже однаковий, незалежно від наявності цієї послуги. Клієнти, які користуються послугою OnlineSecurity мають менший рівень відтоку, ніж ті, хто такої не має, що певно свідчить про хороший користувацький досвід. А також клієнти, які не підписані на сервіс TechSupport, мають тенденцію до відтоку вище, ніж ті, хто користується такою послугою.

Під час аналізу даних важливим також є визначення кореляції між вхідними ознаками і цільовою змінною (відтоком). Визначення кореляції допомагає встановити, які характеристики абонента мають сильний вплив на цільову змінну і які можуть бути менш важливими. Це дозволяє зменшити розмірність простору ознак, виключити незначущі або взаємно залежні ознаки

і покращити ефективність моделювання. З використанням кореляції Пірсона були отримані результати, що зображені на рисунку 3.14.

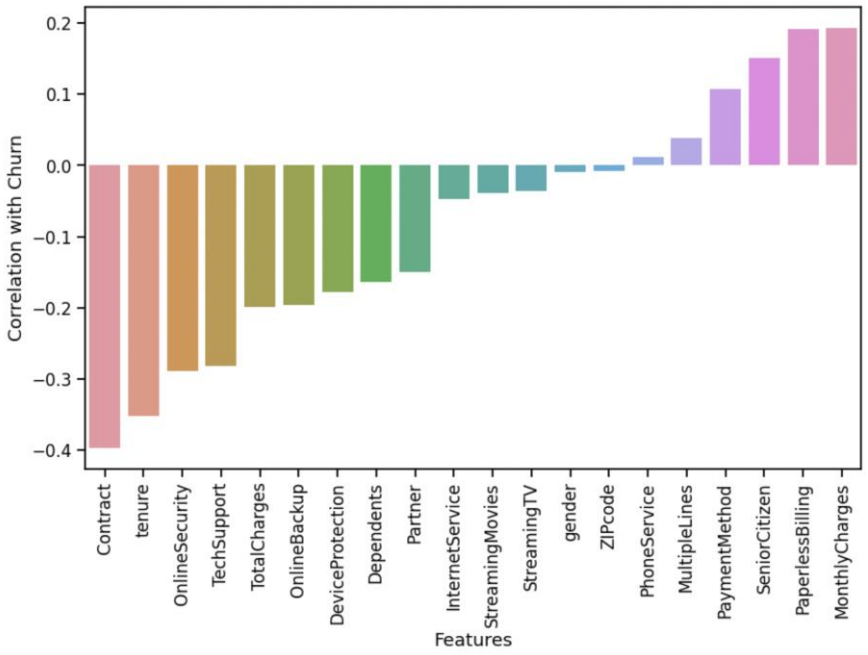


Рисунок 3.14 – Видалення дублікатів

Видалимо ознаку ZIPcode з огляду на низьку кореляцію з цільовою змінною та інформацію, яку вона несе.

3.5 Попередня обробка даних

Як вже було зазначено раніше, процес моделювання вимагає попередньої підготовки даних. Цей процес включає перевірку типів даних і їх конвертацію до одного формату, який найкраще відповідає вимогам обраних моделей та методам аналізу. Перевіримо датасет на наявність викидів, пропущених значень, дублікатів тощо. Також, як вже було зауважено, дані необхідно піддати балансуванню для поліпшення результатів тренування моделей машинного навчання. Окремим важливим етапом є розділення

датасету на тренувальну та тестову вибірки, що дозволить оцінити якість побудованих моделей.

Тож, спершу перевіримо наявність дублікатів та видалимо їх, якщо такі є, цей процес зображений на рисунку 3.15. Дублікати у даних призводить до непотрібного збільшення обсягу вибірки, що займає додаткову пам'ять та ресурси обчислювальної системи. Це може призвести до зниження продуктивності та ефективності обробки даних.

```
if telecom_customers.duplicated().sum() > 0:
    telecom_customers.drop_duplicates(inplace=True)
    print("Duplicates removed")
else:
    print("No duplicates found")
```

```
Duplicates removed
```

```
telecom_customers.shape
```

```
(7021, 20)
```

Рисунок 3.15 – Видалення дублікатів

Також перевіримо датасет на наявність пропущених значень, рисунок 3.16.

```
# Check if we have any NaN values
telecom_customers.isnull().values.any()
```

```
True
```

```
telecom_customers.isnull().sum().to_frame("counts")
```

	counts
gender	0
SeniorCitizen	0
Partner	0
Dependents	0
tenure	0
PhoneService	0
MultipleLines	0
InternetService	0
OnlineSecurity	0
OnlineBackup	0
DeviceProtection	0
TechSupport	0
StreamingTV	0
StreamingMovies	0
Contract	0
PaperlessBilling	0
PaymentMethod	0
MonthlyCharges	0
TotalCharges	11
Churn	0

Рисунок 3.16 – Перевірка на наявність пропущених значень

Тож, атрибут TotalCharges містить пропущені значення. Один з підходів, який використовують в таких ситуаціях, є видалення цих даних, проте не будемо поспішати. Видалення прикладів з пропущеними значеннями може призвести до втрати важливої інформації. Вони можуть містити корисні дані для аналізу та виявлення закономірностей. Іноді розглядають альтернативні варіанти, такі як заповнення пропущених значень середніми або медіанними значеннями. Проте спробуємо інший підхід. Перевіримо чи можна заповнити пропущені значення атрибуту TotalCharges значеннями розрахованими з допомогою MonthlyCharges. Ми знаємо кількість місяців, протягом яких клієнт користується послугами, а також щомісячну плату за них. Проведемо наступні обчислення, спробуємо виразити TotalCharges, через формулу Tenure*MonthlyCharges та подивимось чи мають зміст подібні обчислення для всіх інших прикладів. Отримані результати зображені на рисунку 3.17.

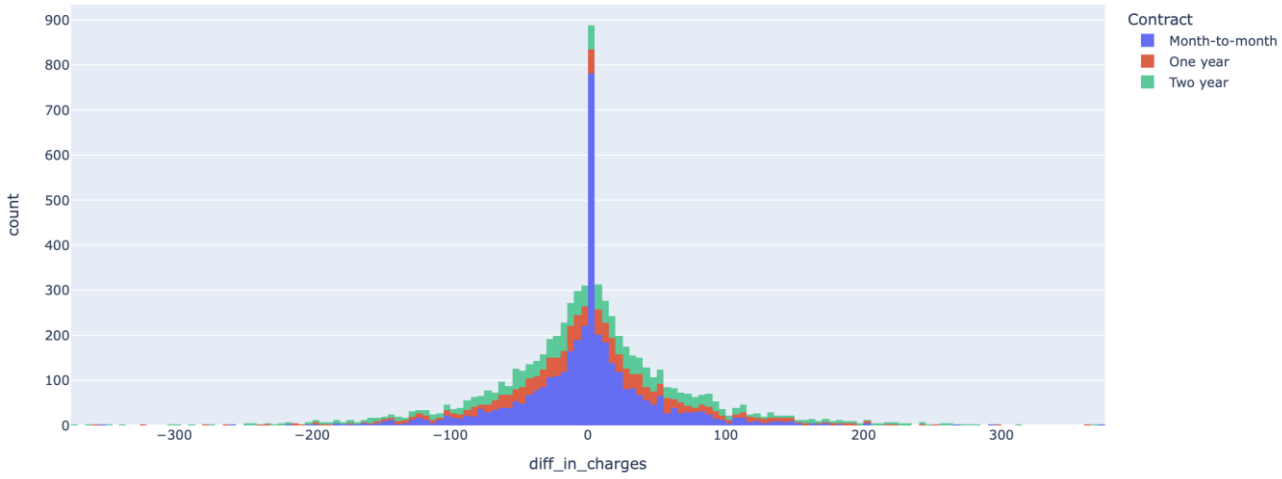


Рисунок 3.17 – Розподіл різниці між фактичними сумарними витратами клієнтів і значеннями, які були обраховані за формулою Tenure*MonthlyCharges

Як бачимо з графіку, що показує розподіл різниці між фактичними сумарними нарахуваннями клієнтів та значеннями, які були обраховані за

міркуваннями вище, ми можемо використати запропонований підхід для заповнення пропущених значень, замість їх видалення.

Проблему з пропущеними даними успішно вирішено, що продемонстровано на рисунку 3.18.

```
# Check if we have any NaN values
telecom_customers.isnull().values.any()
```

```
False
```

Рисунок 3.18 – Перевірка на наявність пропущених значень оновленого датасету

Створимо додаткові змінні, які, можливо, дадуть нам ще більше інформації про відтік користувачів. А саме підрахуємо кількість сервісів, якими користується абонент, а також середню плату за послугу, що буде вираховуватися як загальні витрати користувача поділені на кількість сервісів (рис. 3.19).

Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn	NumberOfServices	TotalAvgChargesPerService
Month-to-month	Yes	Electronic check	29.85	29.85	No	2	14.925000
One year	No	Mailed check	56.95	1889.50	No	4	472.375000
Month-to-month	Yes	Mailed check	53.85	108.15	Yes	4	27.037500

Рисунок 3.19 – Вигляд датасету після додавання нових ознак

Проаналізуємо також унікальні значення ознак вибірки. Як можна побачити з рисунка 3.20, такі змінні як OnlineSecurity, OnlineBackup та деякі інші містять одночасно значення «No internet service» та «No», що фактично дублюється і означає відсутність підписки на сервіс, тож наступним кроком замінимо усі значення «No internet service» на «No». Це спростить обробку та аналіз даних.

```
In column gender there are 2 unique values: ['Female' 'Male']
In column SeniorCitizen there are 2 unique values: [0 1]
In column Partner there are 2 unique values: ['Yes' 'No']
In column Dependents there are 2 unique values: ['No' 'Yes']
In column tenure there are 73 unique values: [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]
In column PhoneService there are 2 unique values: ['No' 'Yes']
In column MultipleLines there are 3 unique values: ['No phone service' 'No' 'Yes']
In column InternetService there are 3 unique values: ['DSL' 'Fiber optic' 'No']
In column OnlineSecurity there are 3 unique values: ['No' 'Yes' 'No internet service']
In column OnlineBackup there are 3 unique values: ['Yes' 'No' 'No internet service']
In column DeviceProtection there are 3 unique values: ['No' 'Yes' 'No internet service']
In column TechSupport there are 3 unique values: ['No' 'Yes' 'No internet service']
In column StreamingTV there are 3 unique values: ['No' 'Yes' 'No internet service']
In column StreamingMovies there are 3 unique values: ['No' 'Yes' 'No internet service']
In column Contract there are 3 unique values: ['Month-to-month' 'One year' 'Two year']
In column PaperlessBilling there are 2 unique values: ['Yes' 'No']
In column PaymentMethod there are 4 unique values: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
In column MonthlyCharges there are 1585 unique values: [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
In column TotalCharges there are 6531 unique values: [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
In column Churn there are 2 unique values: ['No' 'Yes']
In column NumberOfServices there are 9 unique values: [2 4 6 5 7 3 1 9 8]
In column TotalAvgChargesPerService there are 6627 unique values: [ 14.925      472.375       27.0375     ... 173.225      102.2
 977.78571429]
```

Рисунок 3.20 – Унікальні значення ознак

Оскільки більшість з обраних класифікаторів працюють лише з числовими значеннями, нам потрібно перевести категоріальні змінні у числові. Застосуємо Label Encoding для бінарних змінних, надавши їм значення 0/1 залежно від змісту (наприклад Yes-1, No-0). А також використаємо унітарне кодування (One-Hot Encoding), з допомогою функції pd.get_dummies( ) бібліотеки Pandas, для категоріальних змінних з більш ніж двома унікальними значеннями. Основним недоліком такого кодування є значне збільшення розмірності набору даних, тому цього методу слід уникати, коли категоріальний стовпець має велику кількість унікальних значень.

Також корисним для подальшого моделювання буде нормалізація даних, тобто приведення числових стовпців до спільного діапазону значень. Оскільки усі ознаки в наборі даних мають різну природу, вимірюються в різних одиницях, а відповідно мають різні масштаби, це може вплинути на процес навчання та ефективність моделей. Ознаки з вищими значеннями будуть домінувати в процесі навчання, але це не означає, що ці змінні є більш важливими для прогнозування. Нормалізація зможе покращити стабільність і продуктивність алгоритму навчання. Існує багато методів, які можуть в цьому допомогти, в тому числі готові рішення, що пропонують нам різні бібліотеки Python, наприклад StandardScaler(), що є методом з модулю preprocessing бібліотеки scikit-learn.

В даній роботі використовуватиметься Min-Max Normalization, що маштабує ознаки до фіксованого діапазону [0,1] шляхом віднімання мінімального значення ознаки, а потім ділення на діапазон. Процес нормалізації зображено на рисунку 3.21.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Рисунок 3.21 – Процес нормалізації ознак

Тепер дані готові для роботи. Датасет має наступний вигляд (рис. 3.22).

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	Churn	...
0	1	0	1	0	0.013889	0	1	0.115423	0.003437	0	...
1	0	0	0	0	0.472222	1	0	0.385075	0.217564	0	...
2	0	0	0	0	0.027778	1	1	0.354229	0.012453	1	...
3	0	0	0	0	0.625000	0	0	0.239303	0.211951	0	...
4	1	0	0	0	0.027778	1	1	0.521891	0.017462	1	...
...	...	...	...	...	...	...	...	...	...	...	...
7038	0	0	1	1	0.333333	1	1	0.662189	0.229194	0	...
7039	1	0	1	1	1.000000	1	1	0.845274	0.847792	0	...

Рисунок 3.22 – Підготовлений до моделювання набір даних

3.6 Процес моделювання

Перед моделюванням поділимо вибірку. Використовуючи метод train_test_split з бібліотеки scikit-learn, розділимо дані на навчальну та тестову вибірки. Встановимо параметр test_size рівним 0.25, у такому випадку 25% (1756 абонентів) вхідних даних будуть використані для тестування, а 75% (5265 абонентів) - для навчання моделі, а також перемішаємо дані перед поділом встановивши параметр shuffle=True.

Як вже було визначено на етапі аналізу даних, наша навчальна вибірка є незбалансованою, кількість користувачів, що відмовилась від послуг оператора значно менша, ніж кількість тих, що залишились. Було обґрунтовано можливі проблеми, що можуть виникнути під час моделювання,

якщо не врахувати цей факт. Тож використаємо техніки oversampling та undersampling для вирішення проблеми незбалансованих класів.

Використовується метод SMOTE з бібліотеки imbalanced-learn, що є одним з методів oversampling, який створює синтетичні приклади для класу, що складає меншість. sm=SMOTE(random_state=0) створює об'єкт-екземпляр SMOTE, що потім sm.fit_resample(X_train, Y_train) застосовується до навчальних даних X_train і Y_train для збільшення кількості прикладів меншого класу.

Результат, який ми отримали зображений на рисунку 3.23.

```
#oversampling
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 0)
x_sm, y_sm = sm.fit_resample(X_train, Y_train)
```

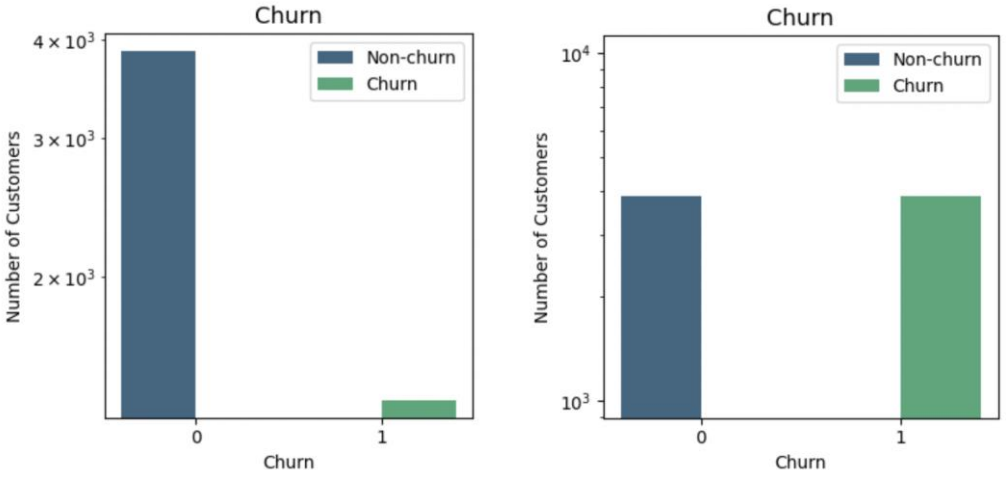


Рисунок 3.23 – Результат балансування класів технікою oversampling

Трохи інша логіка спостерігається при використанні техніки undersampling. RandomUnderSampler з тої ж бібліотеки, використовується для зменшення вибірки, з метою урівнювання класів. Виконується випадкове видалення зайвих прикладів більшого класу.

Результат, який ми отримали зображений на рисунку 3.24.

```
#undersampling
rus = RandomUnderSampler(random_state = 0)
x_rus, y_rus = rus.fit_resample(X_train, Y_train)
```

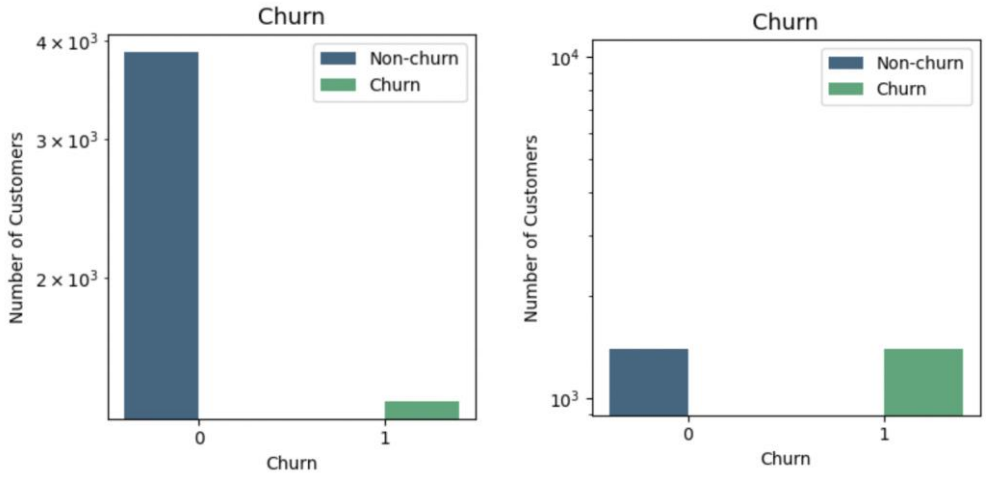


Рисунок 3.24 – Результат балансування класів технікою undersampling

Перейдемо до побудови моделей, що були розглянуті у другому розділі, а саме: логістична регресія, метод опорних векторів, дерева рішень, градієнтний бустинг, наївний баєсівський класифікатор.

3.6.1 Результати моделювання для логістичної регресії

Логістична регресія була першою розглянутою в даній роботі моделлю, для тренування обраної моделі з бібліотеки scikit-learn було імпортовано клас LogisticRegression. Спершу модель було побудовано на даних в їх первинному вигляді, без додаткової обробки та балансування. Також, були застосовані дефолтні параметри моделі, жодні спеціальні налаштування не були використані. Це дає можливість отримати загальне розуміння її можливостей у вирішенні задачі прогнозування відтоку. Аналіз отриманих результатів дозволяє визначити чи необхідні додаткові кроки для поліпшення її прогностичної здатності.

Тож, були отримані наступні результати, рисунки 3.25, 3.26 та 3.27.

	Accuracy	F1 Score	Precision	Recall
Train	0.806838	0.598817	0.667546	0.542918
Test	0.803531	0.594595	0.645408	0.551198

```
--------------------------------------------------------------------------------
Cross-validated Recall scores are: [0.52142857 0.51071429 0.55       0.53763441 0.55197133]
--------------------------------------------------------------------------------
Average Cross-validated Recall score: 0.5343
--------------------------------------------------------------------------------
Cross-validated Recall standard deviation: 0.0161
--------------------------------------------------------------------------------
Actual values    : [1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1]
Predicted values : [1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0]
```

Рисунок 3.25 – Значення метрик якості для моделі логістичної регресії

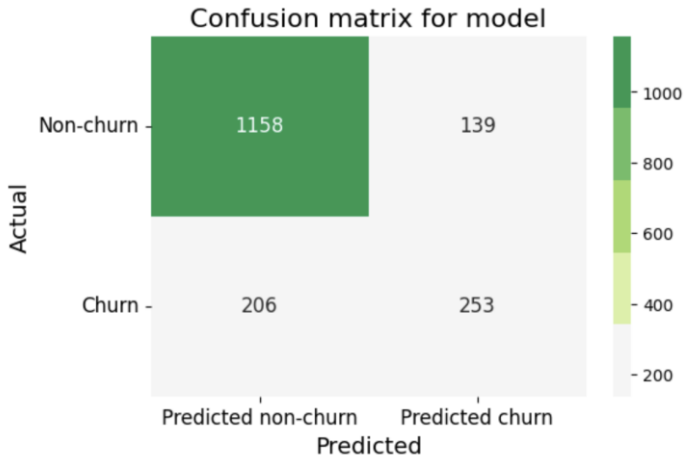


Рисунок 3.26 – Матриця помилок для моделі логістичної регресії

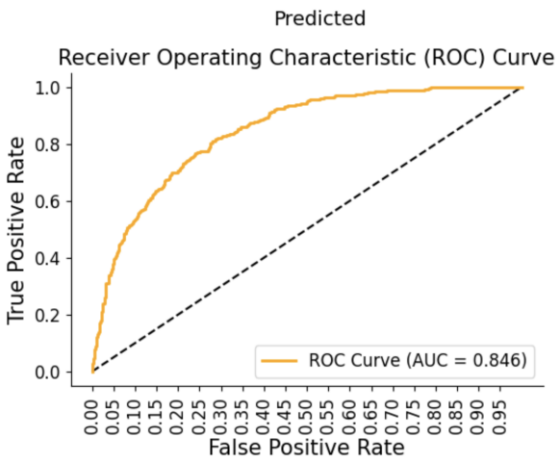


Рисунок 3.27 – Рок крива для моделі логістичної регресії

Бачимо, що у 80% випадках модель класифікує приклади правильно, що є непоганим результатом. Проте, враховуючи природу нашої цільової змінної, для нас важливою є метрика recall, адже вона характеризує здатність моделі виявляти саме позитивний клас, тобто ту категорію людей, що відтікає від компанії. В нашому випадку вона набуває не дуже високого значення.

Ми можемо активного користувача помилково класифікувати як такого, що відтікає. Якщо кількість таких помилок буде великою це проблематично, адже компанія вкладає ресурси в їх утримання, проте це не погіршить лояльність абонентів, хоч і призведе до непотрібних витрат. Але допустити помилку у класифікації тих абонентів, що дійсно мають намір покинути компанію, а класифікатор має погану здатність до їх виявлення, буде коштувати компанії набагато дорожче.

Спробуємо покращити результати класифікації використовуючи збалансовані дані. Отримані результати на рисунках 3.28, 3.29 та 3.30.

	Accuracy	F1 Score	Precision	Recall
Train	0.830230	0.832930	0.81989	0.846393
Test	0.776196	0.613569	0.55914	0.679739

```
--------------------------------------------------------------------------------
Cross-validated Recall scores are: [0.65245478 0.7118863  0.94437257 0.92626132 0.92238034]
--------------------------------------------------------------------------------
Average Cross-validated Recall score: 0.8315
--------------------------------------------------------------------------------
Cross-validated Recall standard deviation: 0.1236
--------------------------------------------------------------------------------
Actual values    : [1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1]
Predicted values : [1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 1]
```

Рисунок 3.28 – Значення метрик якості для моделі логістичної регресії навченої на збалансованих даних

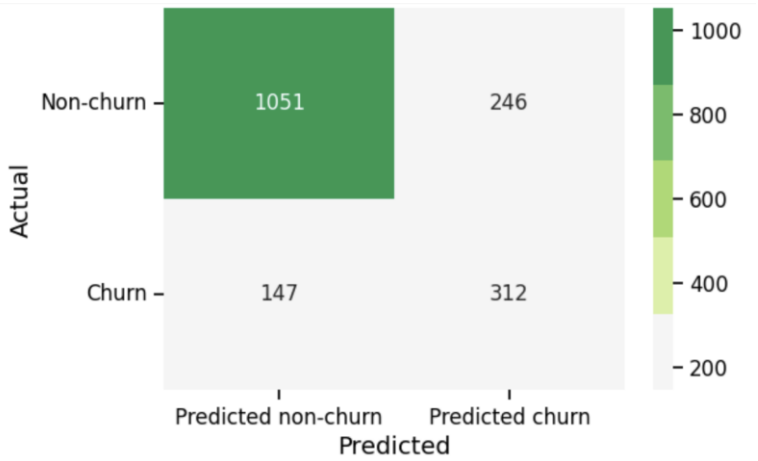


Рисунок 3.29 – Матриця помилок для моделі логістичної регресії навченої на збалансованих даних

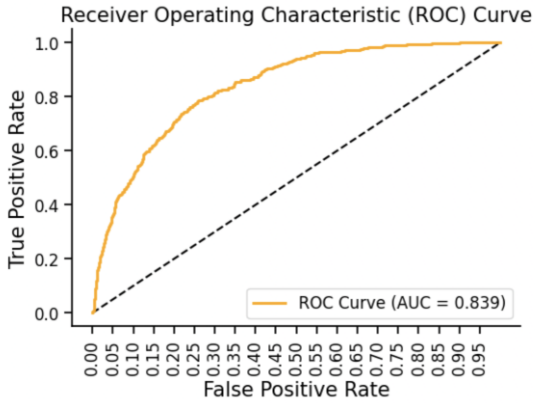


Рисунок 3.30 – Рок крива для моделі логістичної регресії навченої на збалансованих даних

Для збалансованих за допомогою техніки oversampling даних, бачимо покращення значення recall на навчальному наборі даних, проте на тестових даних результати все ще не дуже задовільні. Тож спробуємо використати техніку undersampling, а також встановити параметр solver='saga', цей алгоритм оптимізації підтримує швидку збіжність і використовує стохастичний середньоквадратичний градієнтний спуск (рис.3.31 та 3.32).

	Accuracy	F1 Score	Precision	Recall
Train	0.795558	0.688097	0.572254	0.862745
Test	0.784738	0.668421	0.559471	0.830065

Рисунок 3.31 – Значення метрик якості для моделі логістичної регресії навченої на збалансованих даних

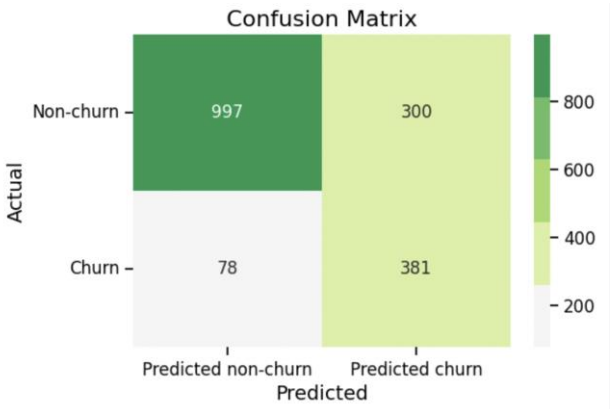


Рисунок 3.32 – Матриця помилок для моделі логістичної регресії навченої на збалансованих даних

Для досягнення ще кращих результатів виконаємо підбір гіперпараметрів за допомогою GridSearchCV (Grid Search Cross-Validation), що виконує перехресну перевірку для оцінки кожної комбінації параметрів моделі. Він розділяє набір даних на кілька фолдів (частин) і проводить тренування та оцінку моделі на кожній комбінації. У налаштуванні моделі будуть брати участь наступні параметри:

1) C: [0.1, 1, 10] – є оберненим значенням параметра регуляризації;

2) penalty: ['l1', 'l2'] – визначає тип регуляризації;

3) solver: ['liblinear', 'saga'] – визначає алгоритм, який використовується для оптимізації функції втрат під час навчання моделі.

Найкраща комбінація та отримані значення метрик і матриця помилок зображені на рисунках 3.33 та 3.34 відповідно.

```
Best Parameters for Logistic Regression: {'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}
```

	Accuracy	F1 Score	Precision	Recall
Train	0.922851	0.801887	0.702479	0.934066
Test	0.903720	0.777251	0.683333	0.901099

Рисунок 3.33 – Значення метрик для моделі логістичної регресії з налаштованими гіперпараметрами

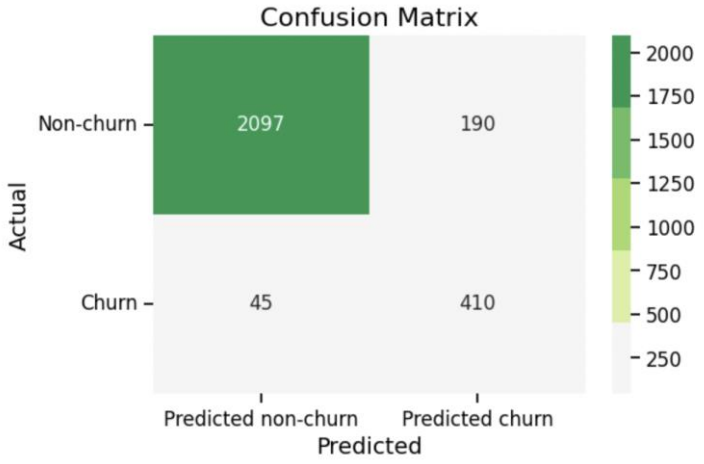


Рисунок 3.34 – Матриця помилок для моделі логістичної регресії з налаштованими гіперпараметрами

Налаштування гіперпараметрів допомогло значно покращити показники якості моделі. Для навчального набору даних маємо досить високий показник точності 0.922851, на тестовому наборі даних точність трохи зменшується до 0.903720, але все ще залишається на високому рівні, що свідчить про хорошу загальну продуктивність моделі.

За допомогою атрибуту .coef_ логістична регресія дозволяє оцінити важливість ознак при прогнозуванні. Кожна ознака має свій коефіцієнт, що відображає, наскільки значущою вона є для прогнозування цільової змінної. Відповідні результати на рисунку 3.35.

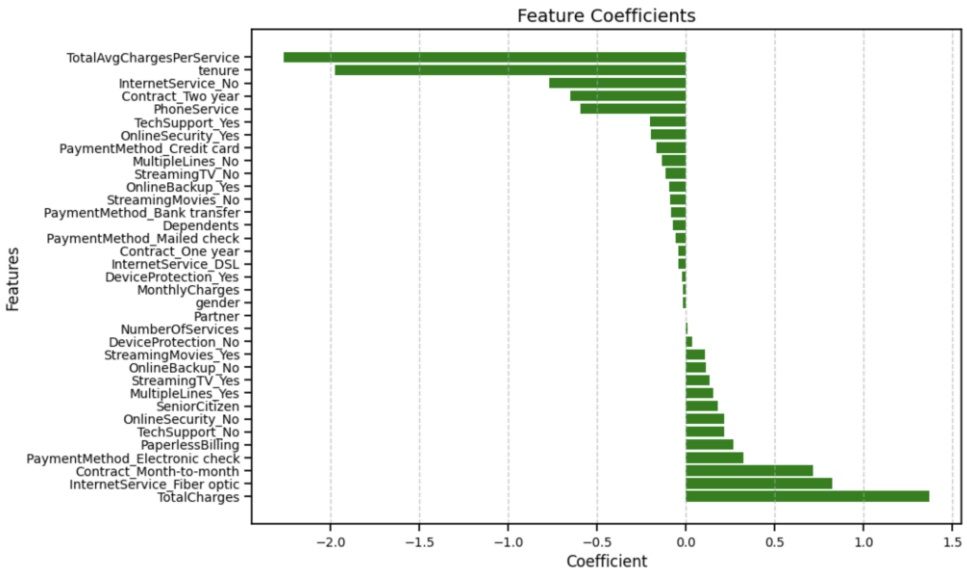


Рисунок 3.35 – Важливість ознак при прогнозуванні логістичної регресії

З отриманого графіку можна помітити, що, наприклад, ознаки tenure та TotalCharges значною мірою впливають на результат прогнозування. Зі збільшенням часу користування послугами (tenure) ймовірність того, що користувач припинить взаємодію з компанією зменшується. А зі збільшенням загальних витрат на послуги оператора (TotalCharges) така імовірність навпаки збільшується. В той же час наявність партнера чи стать клієнта не має великої ваги на результати прогнозування. Отримані результати є досить логічними.

3.6.2 Результати моделювання для дерев рішень

Як і у випадку логістичної регресії спочатку перевіримо ефективність моделі на необробленому наборі даних. Після того використаємо збалансовані різними техніками дані та спробуємо налаштувати параметри моделі для досягнення максимальної продуктивності.

Найкраща точність, а саме 0.73063, на незбалансованих даних була досягнена при використанні критерію джині та максимальної глибини дерева 44. Після застосування збалансованого набору даних були отримані наступні результати, рисунки 3.36, 3.37 та 3.38.

	Accuracy	F1 Score	Precision	Recall
Train	0.998212	0.998209	1.000000	0.996423
Test	0.681663	0.520994	0.429379	0.662309

```
--------------------------------------------------------------------------------
Cross-validated Recall scores are: [0.68571429 0.70967742 0.68817204 0.675      0.68214286]
--------------------------------------------------------------------------------
Average Cross-validated Recall score: 0.6881
--------------------------------------------------------------------------------
Cross-validated Recall standard deviation: 0.0116
```

Рисунок 3.36 – Значення метрик для дерев рішень

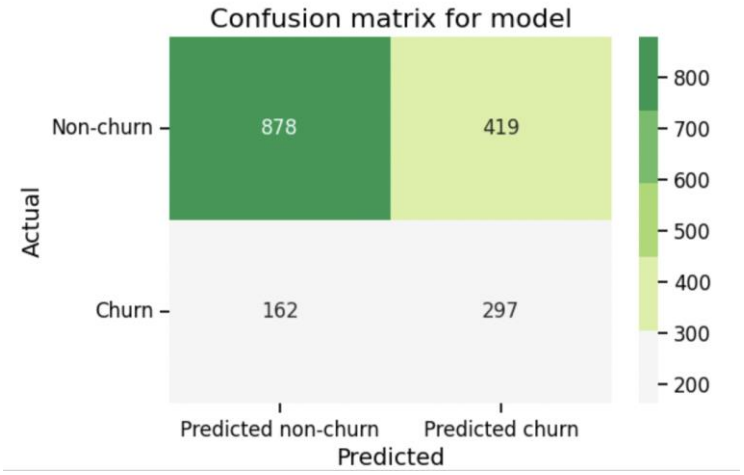


Рисунок 3.37 – Матриця помилок для дерев рішень

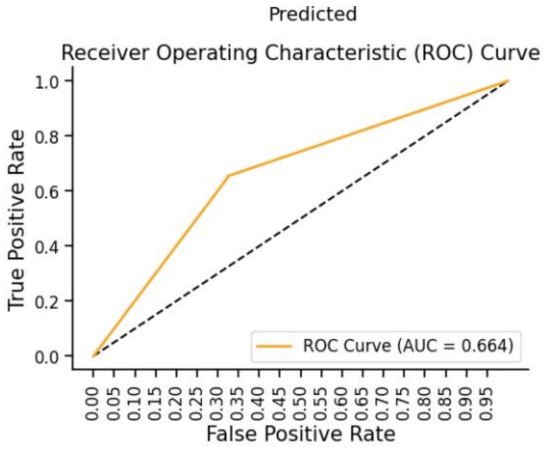


Рисунок 3.38 – Рок крива для дерев рішень

Як бичимо у випадку дерев рішень ми отримали дуже хороші результати на навчальній вибірці та досить низькі показники на тестовій, тож в цій ситуації має місце перенавчання (overfitting). Модель настільки добре «запам’ятала» дані, на яких тренувалась, що не може забезпечити достатньо високу точність на нових, з якими ще не знайома. Тож спробуємо покращити ситуацію виконавши підбір оптимальних параметрів. А саме таких як:

1) criterion: ['gini', 'entropy'] - визначає критерій, за яким відбувається розбиття вузлів дерева під час побудови моделі;

2) max_depth - визначає максимальну глибину дерева;

3) min_samples_split - мінімальна кількість зразків, необхідних для розбиття внутрішнього вузла;

4) min_samples_leaf - визначає мінімальну кількість зразків, яка потрібна для формування листа дерева;

Найкраща комбінація параметрів та результати на рисунках 3.39, 3.40.

Parameters for Decision Tree: {'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 4, 'min_samples_split': 5

	Accuracy	F1 Score	Precision	Recall
Train	0.805180	0.695423	0.601218	0.824635
Test	0.792793	0.673162	0.584615	0.793319

Рисунок 3.39 – Метрики якості для дерев рішень з налаштованими гіперпараметрами

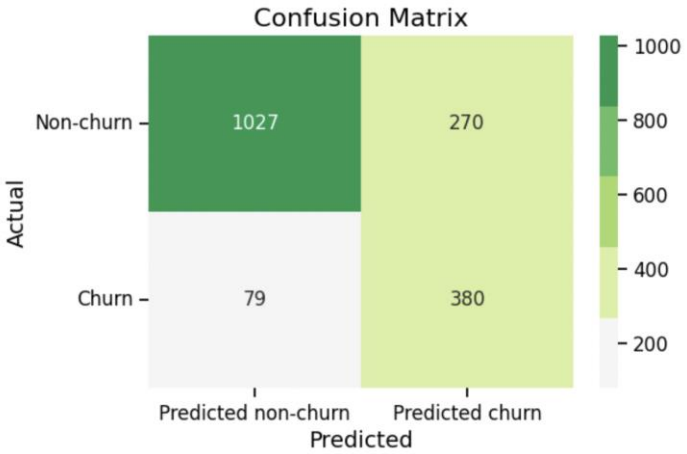


Рисунок 3.40 – Матриця помилок для дерев рішень з налаштованими гіперпараметрами

Загальні результати покращились, в 80% випадків модель дає правильну відповідь, проте нас цікавить класифікатор з найкращими результатами, тому продовжимо дослідження.

Розглянемо також атрибут feature_importances_, який є доступним для моделі, що розглядається, і вказує наскільки сильно кожна ознака впливає на прийняття рішень моделлю під час прогнозування. Відповідний графік на рисунку 3.41.

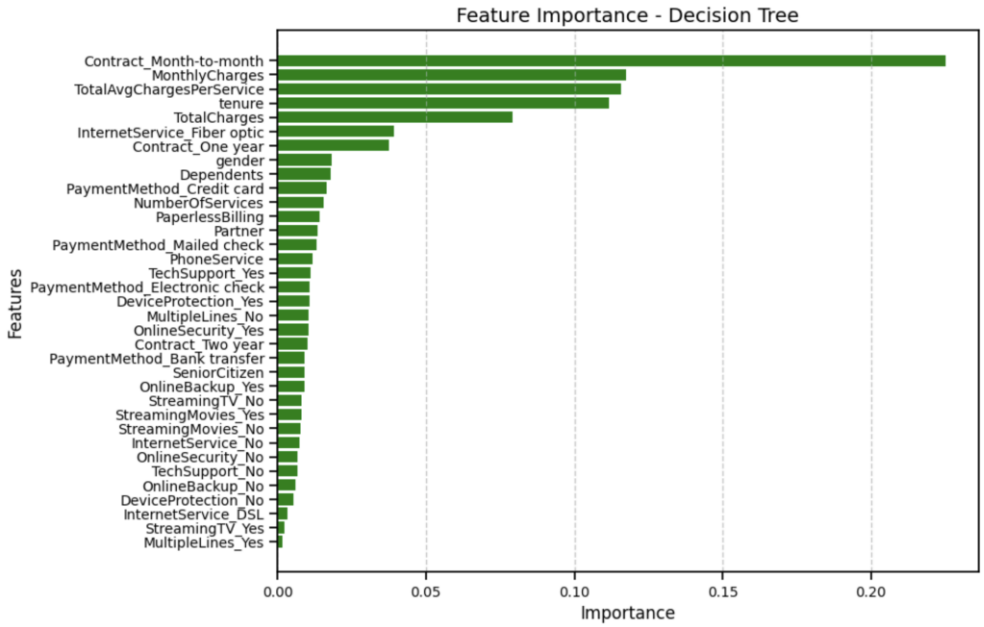


Рисунок 3.41 – Важливість ознак при прогнозуванні з використанням дерев рішень

3.6.3 Результати моделювання для наївного баєсівського класифікатора

Досить непогані результати показав наївний баєсівський класифікатор. На початкових даних маємо наступні результати, рисунки 3.42, 3.43, 3.44.

	Accuracy	F1 Score	Precision	Recall
Train	0.756795	0.765517	0.739015	0.793991
Test	0.741458	0.620401	0.503392	0.808279

```
--------------------------------------------------------------------------------
Cross-validated Recall scores are: [0.81071429 0.78494624 0.83512545 0.73928571 0.79285714]
--------------------------------------------------------------------------------
Average Cross-validated Recall score: 0.7926
--------------------------------------------------------------------------------
Cross-validated Recall standard deviation: 0.0317
```

Рисунок 3.42 – Значення метрик для наївного баєсівського класифікатора

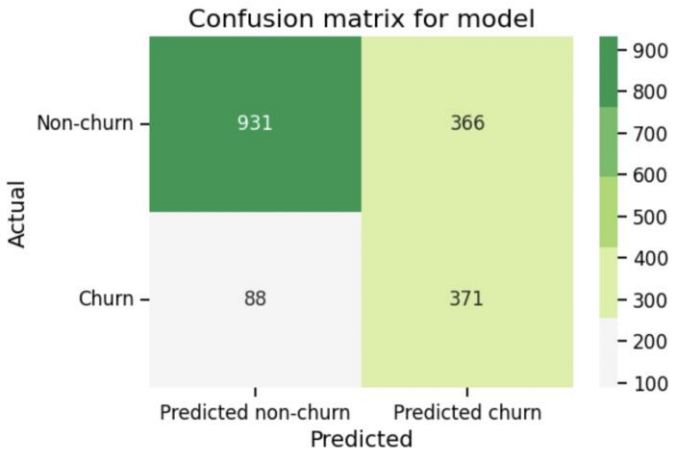


Рисунок 3.43 – Матриця помилок для наївного баєсівського класифікатора

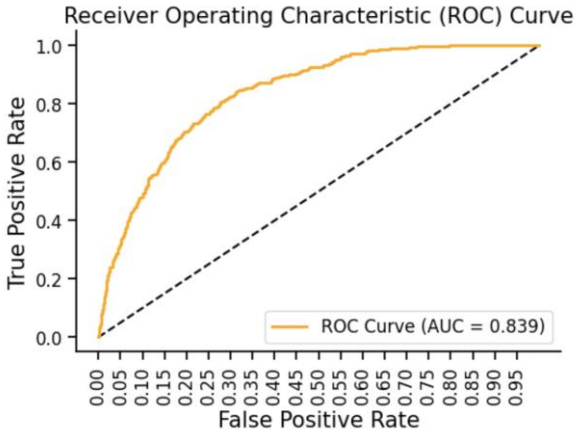


Рисунок 3.44 – Рок крива для наївного баєсівського класифікатора

Далі модель тренувалася на вже збалансованих даних, окрім того було застосовано ще один підхід до покращення якості моделі. У пункті 3.3 було розглянуто кореляцію Пірсона, що вимірює лінійну залежність між двома змінними, в нашому випадку між цільовою змінною та ознаками. Для деяких змінних отримані коефіцієнти були близькі до нуля, у такому випадку ці ознаки, ймовірно, не мають суттєвого впливу на передбачення цільової змінної і можуть бути видалені. Результати після видалення ознак на рисунках 3.45, 3.46.

	Accuracy	F1 Score	Precision	Recall
Train	0.942483	0.894901	0.856574	0.936819
Test	0.901481	0.864754	0.816248	0.919390

Рисунок 3.45 – Значення метрик для наївного баєсівського класифікатора

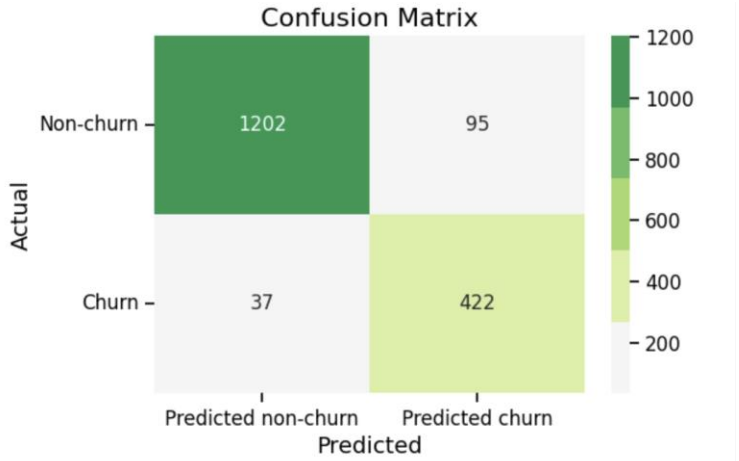


Рисунок 3.46 – Матриця помилок для наївного баєсівського класифікатора

3.6.4 Результати моделювання для методу опорних веткорів

Для даного методу машинного навчання проводились подібні дослідження. Навчена на збалансованих даних модель з встановленим параметром kernel='linear' показала не дуже високу точність, проте досить хорошу здатність розпізнавати людей, що є на межі відтоку, рисунок 3.47 та 3.48.

	Accuracy	F1 Score	Precision	Recall
Train	0.740701	0.762061	0.704063	0.830472
Test	0.699317	0.594470	0.459075	0.843137

Рисунок 3.47 – Значення метрик для метода опорних векторів

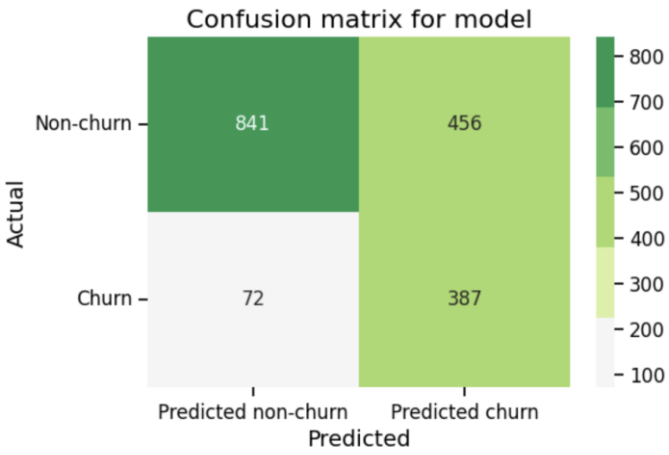


Рисунок 3.48 – Матриця помилок для методу опорних векторів

Спробуємо покращити модель налаштуванням гіперпараметрів, найкращі результати були отримані при комбінації: 'C': 10, 'gamma': 0.001, 'kernel': 'rbf'. Результати моделі на рисунках 3.49, 3.50.

	Accuracy	F1 Score	Precision	Recall
Train	0.962984	0.930481	0.913866	0.947712
Test	0.928246	0.918977	0.899791	0.938998

Рисунок 3.49 – Значення метрик для метода опорних векторів з налаштованими гіперпараметрами

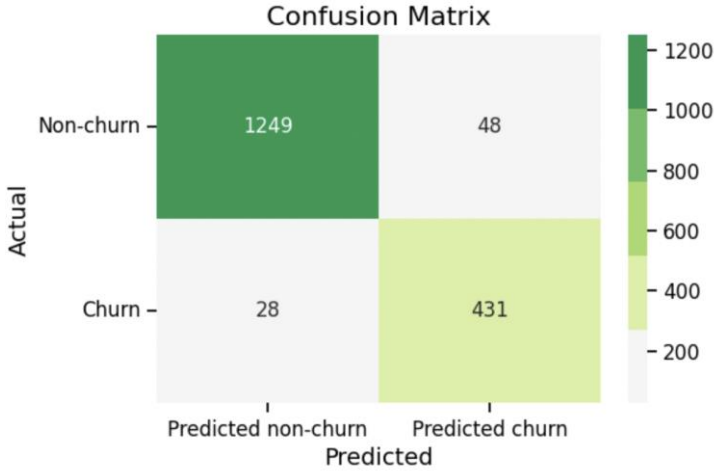


Рисунок 3.50 – Матриця помилок для метода опорних векторів з налаштованими гіперпараметрами

3.6.5 Результати моделювання для XGBClassifier

XGBClassifier з бібліотеки XGBoost виявився дуже ефективним класифікатором в умовах даної задачі, з високою продуктивністю та точністю. Після проведеної роботи над моделлю та налаштування гіперпараметрів найкращі результатами дала наступна комбінація: 'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 200, 'subsample': 0.9. Результати моделі на рисунках 3.51, 3.52.

	Accuracy	F1 Score	Precision	Recall
Train	0.965831	0.936170	0.914761	0.958606
Test	0.959567	0.924548	0.902490	0.947712

Рисунок 3.51 – Значення метрик для XGBClassifier з налаштованими гіперпараметрами

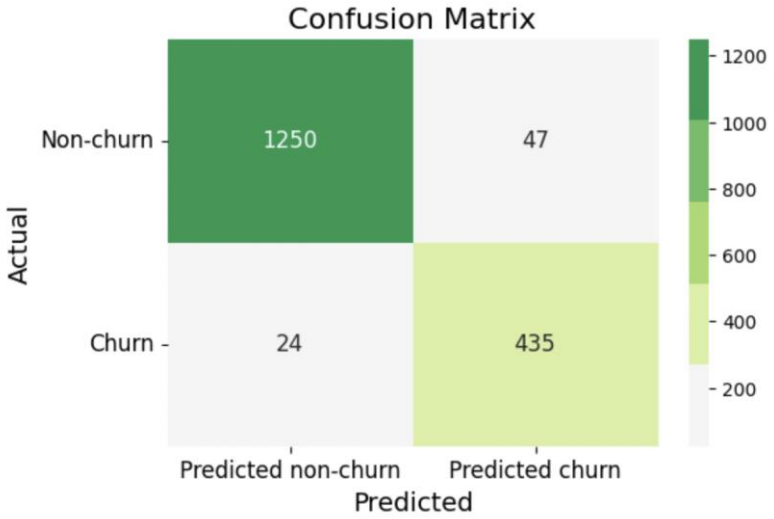


Рисунок 3.52 – Матриця помилок для XGBClassifier з налаштованими гіперпараметрами

3.7 Висновки до розділу 3

Для зручності найкращі результати по кожній моделі були зведені в одну таблицю 3.1.

Таблиця 3.1 – Порівняння моделей прогнозування

Model	Accurancy	F1 Score	Precision	Recall
Logistic Regression	0.903720	0.777251	0.68333	0.901099
SVM	0.928246	0.918977	0.899791	0.938998
Naive Bayes	0.901481	0.864754	0.816248	0.919390
Decision Tree	0.792793	0.67162	0.584615	0.793319
XGBClassifier	0.959567	0.924548	0.902490	0.947712

Для зручного порівняння моделей візуалізуємо отримані результати, що зображено на рисунках 3.53 та 3.54.

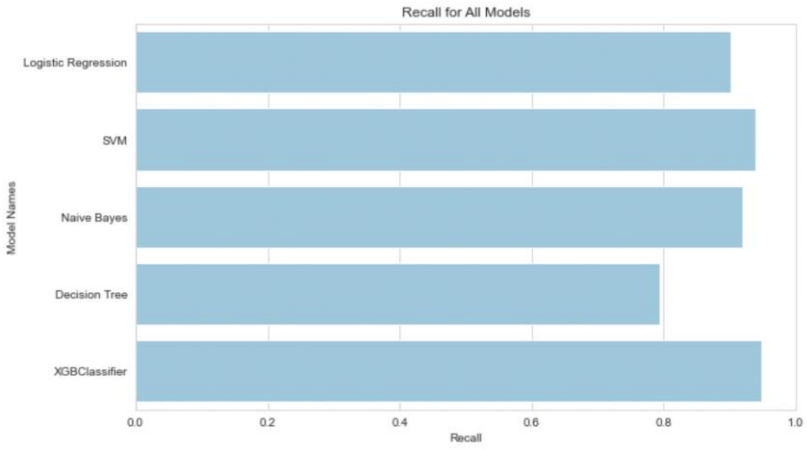


Рисунок 3.53 – Порівняння використаних моделей машинного навчання на основі метрики Recall

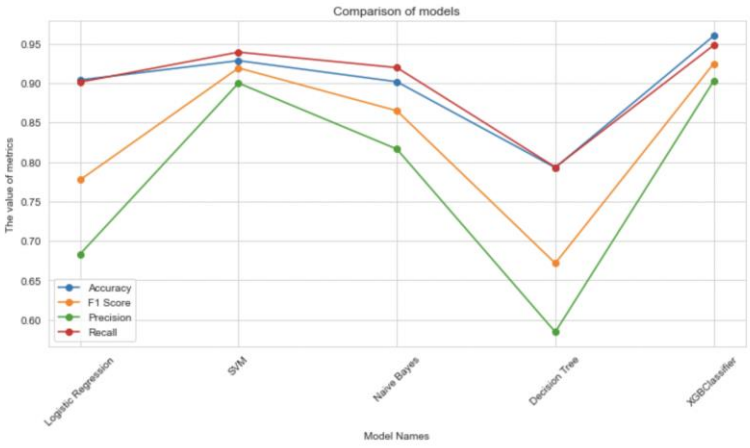


Рисунок 3.54 – Порівняння використаних моделей машинного навчання

Усі моделі показали досить високу якість, проте найкращими виявилися XGBClassifier та SVM, окрім високої точності вони мають найвищі показники Recall, що дозволяє прогнозувати відтік користувачів та приймати відповідні стратегічні рішення для збереження аудиторії.

Хоча XGBClassifier та SVM є відомими алгоритмами машинного навчання, використання їх у даній роботі має свою унікальність. Дослідження спрямоване на вирішення конкретної проблеми в телекомунікаційній галузі - прогнозування відтоку користувачів. Використання відомих моделей машинного навчання в цьому контексті вимагає додаткової настройки та адаптації до специфічних вимог телекомунікаційного сектора. А будь який дослідницький процес включає послідовність методів, аналіз даних, а також вимагає прийняття рішення на кожному етапі, що дає можливість отримати оптимальний алгоритм для забезпечення відповідної точності прогнозування.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі буде проведено оцінювання основних характеристик розробленого програмного продукту, а саме системи інтелектуального аналізу даних для прогнозування відтоку аудиторії в телекомунікаційному секторі.

Дана реалізація має свою цінність не тільки в Україні, а й поза її межами, оскільки телекомунікаційна галузь є однією з провідних в економіці будь-якої країни, а ефективний інструмент управління відтоком сприятиме підтримці та розвитку бізнесу в цьому секторі.

В межах даного дослідження розглядаються різні варіанти реалізації системи для забезпечення її економічної ефективності та вибору оптимальної стратегії. З цією метою буде задіяно апарат функціонально-вартісного аналізу, що дозволяє оцінити реальну вартість продукту або послуги, незалежно від організаційної структури компанії. ФВА є ефективним інструментом виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, оптимального співвідношення між споживчою вартістю продукту та витратами на його розробку. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі використовується метод функціонально-вартісного аналізу (ФВА) для проведення техніко-економічного аналізу системи прогнозування стійкості фінансових показників. Кожна окрема підсистема повинна

відповідати вимогам усієї системи, оскільки рішення, прийняті щодо проектування та розробки компонентів, мають безпосередній вплив на функціонування самої системи.

Тож, фактичний аналіз передбачає аналіз функцій програмного продукту, що використовуються для збору, обробки та аналізу даних по компанії.

Технічні вимоги до програмного продукту наступні.

1. Функціонування на персональних комп’ютерах із стандартним набором компонентів. Програмний продукт має бути сумісним з популярними операційними системами, такими як Windows, macOS і Linux. А також підтримувати стандартні протоколи та інтерфейси, які використовуються на персональних комп'ютерах, наприклад, TCP/IP для мережевого зв'язку.

2. Зручність та зрозумілість для користувача. Програмний продукт повинен бути інтуїтивно зрозумілий інтерфейс, що дозволить кінцевому користувачу швидко оволодіти його функціоналом.

3. Швидкість обробки даних та доступ до інформації в реальному часі. Програмний продукт має забезпечувати оптимальну продуктивність обробки даних та здатність оперативно опрацьовувати великі обсяги даних.

4. Можливість зручного масштабування та обслуговування. Архітектура програмного продукту повинна бути гнучкою та розширюваною, що дозволить легко додавати нові функції та модулі.

5. Мінімальні витрати на впровадження програмного продукту. Вимоги до ресурсів повинні бути мінімізовані, але забезпечувати повну працездатність програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F0 – розробка можливого програмного продукту, що дозволяє аналізувати різні характеристики, які безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

- F1 – вибір мови програмування;
- F2 – якісний аналіз даних та впровадження нових атрибутів;
- F3 – методи реалізації моделей.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F1 :

А) Python

Б) C++

Функція F2 :

А) Проведення аналізу та обробку даних

Б) Використання даних без попередньої обробки

Функція F3 :

А) Використання готових модулів

Б) Побудова власних алгоритмів

Варіанти реалізації основних функцій наведені у морфологічній карті системи, рисунок 4.1. На основі цієї карти побудовано позитивно- негативну матрицю варіантів основних функцій, таблиця 4.1.

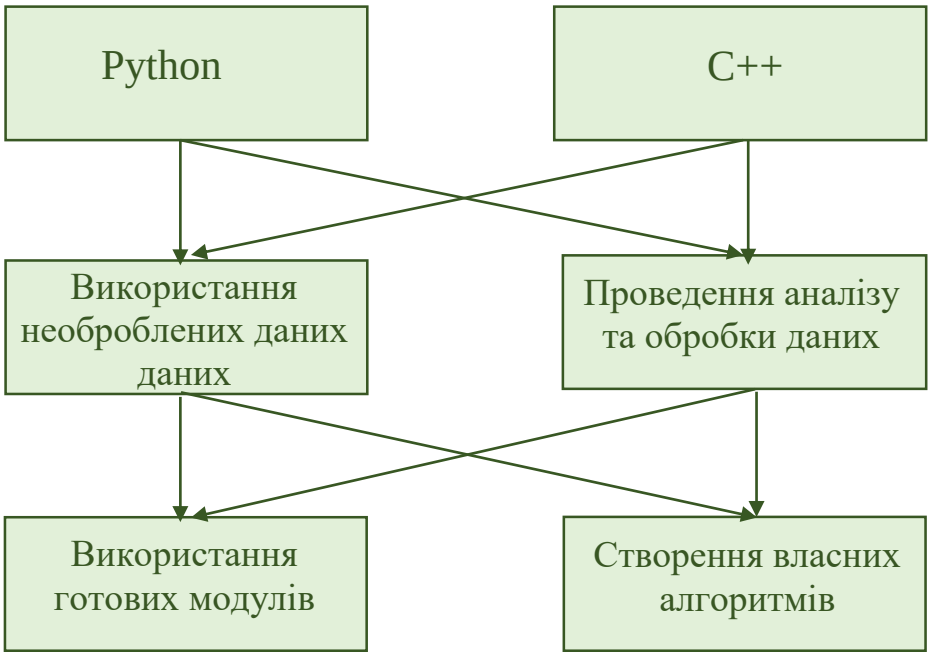


Рисунок 4.1 – Морфологічна карта

Морфологічна карта наочно показує можливі комбінації основних функцій, що можуть бути використані в програмному продукті. Кожна комбінація є унікальною і представляє можливий підхід до використання функцій для досягнення поставлених цілей. Морфологічна карта допомагає визначити набір функцій, які найкраще відповідають потребам проекту.

Таблиця 4.1 відображає оцінку комбінацій функцій за позитивними та негативними аспектами у формі позитивно-негативної таблиці.

Таблиця 4.1 - Позитивно-негативна матриця

Функція	Варіанти реалізації	Переваги	Недоліки
F1	А	Простота використання, високий рівень інтегро-ваності та наявність ефективних бібліотек	Низька швидкість роботи, особливо з великою кількістю даних
	Б	Висока швидкість виконання коду	Вимагає більше часу на етапі розробки програмного продукту
F2	А	Збереження контексту, що не залежить від прийнятих на етапі обробці рішень, а також економія часового ресурсу	Потенційні витрати від помилок системи
	Б	Можливість збільшити точність системи	Додаткові витрати фінансів, ресурсів та часу
F3	А	Нижчий рівень необхідних компетенцій та значна економія часового ресурсу	Менший простір до кастомізації та обмежена настроюваність
	Б	Краща адаптивність до конкретної задачі та предметної області	Вимагає більше часу та компетенцій на етапі розробки програмного продукту

Тож, функція F1 має два варіанти реалізації. Варіант А відрізняється простотою використання, високим рівнем інтегрованості та наявністю стандартних бібліотек, що забезпечують високу ефективність та швидкість навчання, проте при роботі з великими об’ємами даних може зменшуватись швидкодія. Варіант Б натомість може забезпечити високу швидкість коду, проте вимагає більше часу на етапі розробки програмного продукту. Вибір між цими варіантами залежить від потреб користувача, проте в даній роботі пріоритет був наданий варіанту А з огляду на перелічені вище переваги рішення.

Функція F2 також пропонує два альтернативні варіанти. Точність системи можна підвищити за допомогою варіанта Б, але це вимагає додаткових витрат ресурсів, часу та бюджету. Варіант А, натомість, може пришвидшити впровадження рішення, а також зменшити витрати ресурсів. Вибір між цими стратегіями залежить від компромісу між точністю та витратами на вдосконалення системи. Проте обидва варіанти мають місце в процесі розробки.

Функція F3 також має два варіанти реалізації. Варіант А має перевагу в швидкості та вимагає нижчого рівня компетенцій, що є перевагою. Проте цей варіант пропонує менший простір для кастомізації. Варіант Б, натомість, забезпечує кращу адаптивність до конкретної задачі та предметної області, але вимагає більше ресурсів на етапі розробки програмного продукту. Вибір між цими варінатами залежить від специфічних вимог та можливостей фінансування проекту.

Отже, вибір оптимальних варіантів залежить від маштабу проєкту, доступних ресурсів, вимог до гнучкості і контролю над розробкою. Можна побачити, що деяких варіантів реалізації функцій слід уникати при створенні програмного продукту, зважаючи на обґрунтовані вище переваги та недоліки. Остаточний вибір оптимальної комбінації варіантів реалізації буде здійснено на основі зважених критеріїв, що забезпечить компроміс між поточними потребами системи та доступними ресурсами.

1. F1 - Обрано варіант А, Python, оскільки дана мова пропонує більшу інтегрованість, що є необхідістю для розробки значних систем.

2. F2 - Може бути використано як А, так і Б варіант.

3. F3 - Обрано варіант А, оскільки він забезпечує оптимальну швидкість процесу розробки та ефективне функціонування системи.

Отже, будемо розглядати наступні комбінації варіантів реалізації ПП:

1. F1(А) - F2(А) - F3(А)
2. F1(А) - F2(Б) - F3(А)

4.3 Вибір параметрів для програмного продукту

Основні параметри вибору встановлюються на основі проведеного аналізу та будуть використані для обчислення коефіцієнта технічного рівня програмного продукту. Під час проведення оцінки програмного продукту будуть враховуватися наступні параметри:

1) X1 - швидкість мови програмування;
2) X2 - об'єм пам'яті, необхідний зберігання даних;
3) X3 - час попередньої обробки даних;
4) X4 - очікуваний об’єм програмного коду.
5) $X1$: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Вказує на розмір програмного коду, необхідного для створення продукту безпосередньо розробнику.

Значення вищезазначених параметрів (добре, посередньо, погано) будуть визначатися на основі потреб замовника та умов експлуатації програмного продукту. Для кожного параметра встановлюються критерії оцінювання, які допомагають оцінити виконання вимоги та визначити якого рівня якості можна досягнути. Детальніше у таблиці 4.2.

Таблиця 4.2 - Параметри програмного продукту

Параметр	Позначення	Одиниці виміру	Значення		
			Погано	Посередньо	Добре
Швидкодія	X1	Операція/мс	40	100	200
Об'єм пам'яті	X2	Мб	32	16	8
Час попередньої обробки даних	X3	Мс	600	450	70
Очікуваний обсяг програмного коду	X4	Кількість рядків коду	1600	1000	800

На основі таблиці вище, було побудовано графіки для кожного параметра. Детальніше на рисунках 4.2, 4.3, 4.4 та 4.5.

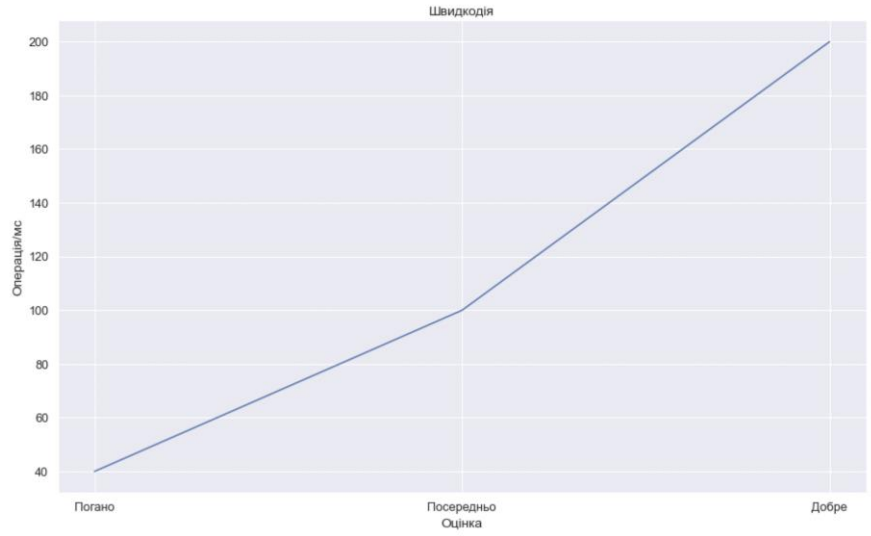


Рисунок 4.2 – X1 (швидкість мови програмування)

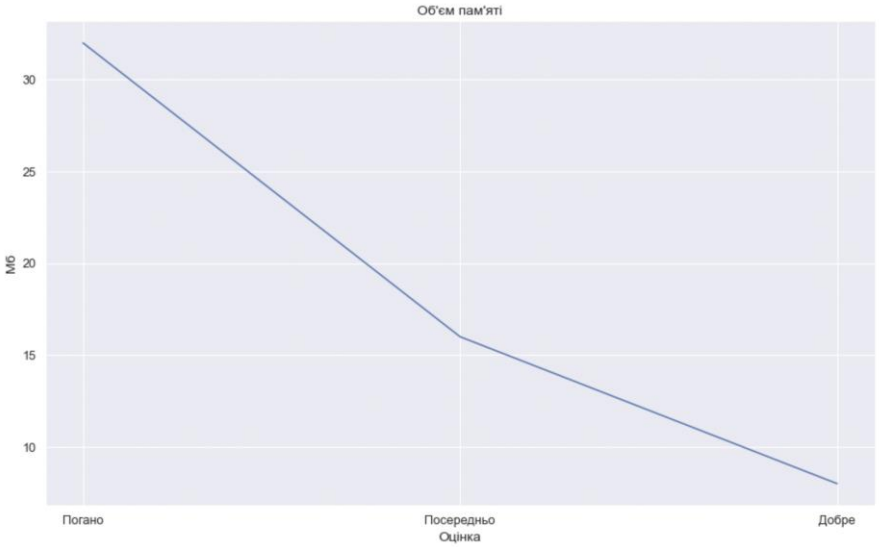


Рисунок 4.3 – X2 (об’єм пам’яті)

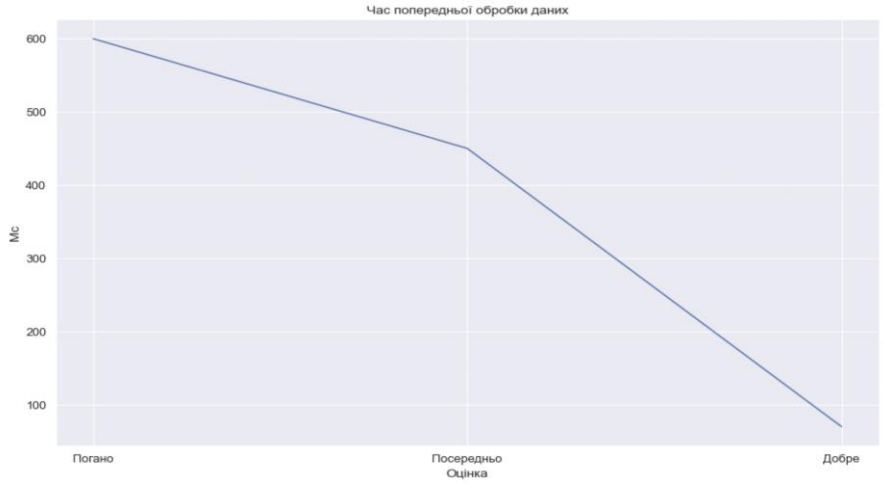


Рисунок 4.4 – X3 (час попередньої обробки даних)

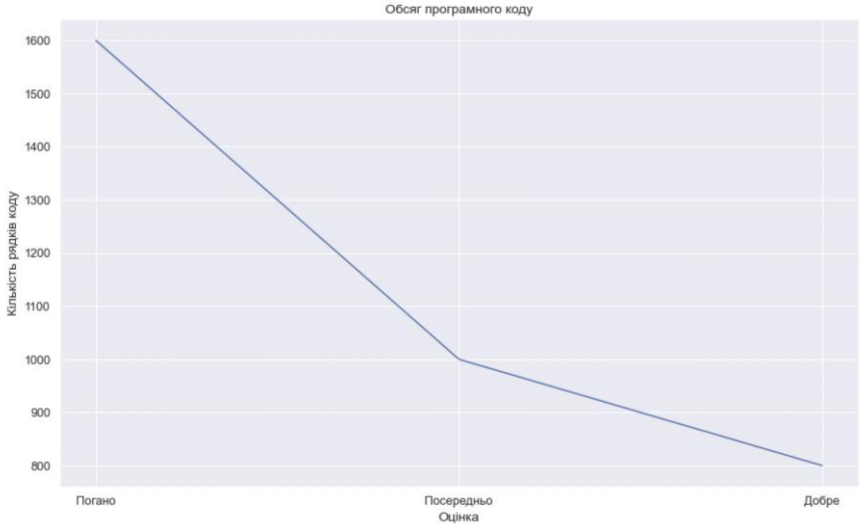


Рисунок 4.5 – X4 (потенційний об’єм програмного коду)

4.4 Експертний аналіз параметрів

Після детального аналізу та вивчення кожного параметра експерт визначаєи значення кожного параметра для створення програмного забезпечення, яке забезпечить найбільш точні результати моделювання адаптивного прогнозування та обчислення прогнозної вартості. Для визначення значимості кожного параметра використовується підхід попарного порівняння. Оцінку проводить експертна комісія, що складається з 7 осіб. Процес визначення коефіцієнтів значимості для кожного параметра складається з декількох етапів, а саме встановлення рівня значимості параметра, перевірка придатності експертних оцінок, визначення попарного пріоритету параметрів та обробка результатів для визначення коефіцієнта значимості (табл.4.3).

Таблиця 4.3 - Результати оцінки параметрів від експертів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія	Опер./мс	3	4	4	3	3	4	3	24	6,5	42,25
X2	Об'єм пам'яті	Mb	2	2	1	2	1	1	2	11	-6.5	42,25
X3	Час попередньої обробки даних	Мс	1	1	2	1	2	2	1	10	-7,5	56,25
X4	Потенційний обсяг програмного коду	Кількість рядків коду	4	3	3	4	4	3	4	25	7,5	56,25
	Сума		10	10	10	10	10	10	10	70	0	197

Для перевірки достовірності експертних оцінок визначимо наступні параметри. Обчислимо суму рангів кожного з параметрів. Для цього додамо всі ранги, які були присвоєні експертами кожному параметру.

Також обчислимо загальну суму рангів, використовуючи формулу (4.1), яка дозволяє знайти загальну суму рангів.

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70 \qquad (4.1)$$

де N - кількість експертів, n - кількість параметрів.

Обчислимо середню суму рангів за формулою (4.2):

$$T = \frac{1}{n} R_{ij} = 17.5 \quad (4.2)$$

Обчислимо відхилення суми рангів кожного параметра від середньої суми рангів за формулою (4.3):

$$\Delta_i = R_i - T \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0.

Обчислимо загальну суму квадратів відхилень за формулою (4.4):

$$S = \sum_{j=1}^{N} \Delta_i^2 = 197 \quad (4.4)$$

Порахуємо коефіцієнт узгодженості за формулою (4.5):

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 197}{7^2(4^3-4)} = 0.804 > W_k = 0.67 \quad (4.5)$$

де n - кількість параметрів, в данному випадку n = 4.

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	<	>	<	<	<	>	<	<	0,5
X2 і X3	>	>	<	>	<	<	>	>	1,5
X2 і X4	<	<	<	<	<	<	<	<	0,5
X3 і X4	<	<	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі (4.6).

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складається матриця $A=\| a_{ij} \|$.

Для кожного параметра розраховується вагомість $K_{\text{ві}}$ згідно з наступними формулами (4.7) та (4.8).

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів, поки наступні значення не відрізняються від попередніх менше ніж на 2%. На наступних кроках відносні оцінки розраховуються за наступними формулами (4.9) та (4.10).

$$K_{\text{ві}} = \frac{b'_i}{\sum_{i=1}^{n} b'_i}, \quad (4.9)$$

$$b'_i = \sum_{i=1}^{N} a_{ij}\, b_j \quad (4.10)$$

З таблиці 4.5 видно, що різниця між значеннями коефіцієнтів вагомості не перевищує 2%, тому додаткові ітерації не потрібні.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	bi	Кві	bi1	Кві1	bi2	Кві2
X1	1	1,5	1,5	0.5	4.5	0,28	16,25	0.28	59.125	0,27
X2	0,5	1	1,5	0,5	3.5	0,22	12,25	0,21	44.875	0,21
X3	0,5	0,5	1	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X4	1.5	1,5	1,5	1	5.5	0,34	21,25	0,35	77,875	0,36
Всього					16	1	59	1	216	1

4.5 Аналіз якості реалізації варіантів функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Розглядаються абсолютні значення різних параметрів: X2 (Об’єм пам’яті), X3 (час попередньої обробки даних) та X4 (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП. Абсолютне значення параметра X1 (швидкість роботи мови програмування) обрано не найгіршим.

Для визначення показників якості кожної варіації реалізації програмного продукту використовується коефіцієнт технічного рівня. Усі розрахунки занесені в таблицю 4.6, в кожен показник якості визначається за формулою (4.11), яка множить вагу параметра на загальну суму балів параметра.

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}i,j} B_i, j \qquad (4.11)$$

де n - кількість параметрів;

К*в*i,j - коефіцієнт вагомості і-го параметра;

Вi,j - оцінка і-го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	Х1	110	6	0,27	1,62
F2	А	Х2	16	6	0,21	1,26
	Б	Х3	300	9	0,16	1.44
F3	А	Х4	900	5	0,36	1.8

За даними з таблиці 4.6 за формулою:

$$К_К = К_{ТУ}[F_{1k}] + К_{ТУ}[F_{2k}] + \cdots + К_{ТУ}[F_{zk}] \qquad (4.12)$$

Визначаємо рівень якості кожного з варіантів:

$$К_{К1} = 1,62 + 1,26 + 1.8 = 4.68$$

$$К_{К2} = 1,62 + 1.44 + 1.8 = 4.86$$

Як видно з розрахунків, кращим виявився другий варіант, для якого коефіцієнт технічного рівня має найбільше значення

4.6 Економічний аналіз розробки

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Кожен варіант складається з двох окремих завдань:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, а завдання 2 — до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1(алгоритми оптимізації та моделювання систем і об’єктів); а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних. Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \qquad (4.13)$$

де K_P - трудомісткість розробки ПП,

$K_П$ - поправочний коефіцієнт,

$K_{СК}$ - коефіцієнт на складність вхідної інформації,

K_M - коефіцієнт рівня мови програмування,

$K_{СТ}$ - коефіцієнт використання стандартних модулів і прикладних програм,

$К_{СТ.М}$ - коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ = 90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_п$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $К_{ск}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{ст}$ = 0.8 для 1 варіанту реалізації. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$$Т_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (122.4 + 19.44 + 4.8 + 10.8) \cdot 8 = 1328.64 \text{ людино-годин;}$$

$$Т_{II} = (122.4 + 19.44 + 6.91 + 10.8) \cdot 8 = 1345.52 \text{ людино-годин.}$$

Як видно з обчислень, другий варіант має більшу трудомісткість.

Для розробки програмного продукту залучається один програміст з окладом 32 000 грн та один спеціаліст з обробки даних телекомунікаційного сектора з окладом 28 000 грн. Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{Т_m \cdot t} \text{грн,} \qquad (4.14)$$

де М - місячний оклад працівників,

Tm - кількість робочих днів у місяць,

t - кількість робочих годин в день.

$$С_Ч = \frac{32\,000 + 28\,000}{3 \cdot 21 \cdot 8} = 119.048 \text{ грн.}$$

Тоді розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \qquad (4.15)$$

де $С_Ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 119.048 \cdot 1328.64 \cdot 1.2 = 189\,806{,}32$ грн.

II. $С_{ЗП} = 119.048 \cdot 1345.52 \cdot 1.2 = 192\,217{,}77$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 189\,806{,}32 \cdot 0.22 = 41757.39$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 192\,217{,}77 \cdot 0.22 = 42287{,}9094$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 32 000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 32\,000 \cdot 0{,}2 = 76\,800 \text{ грн.}$$

З урахуванням додаткової заробітної плати, отримуємо загальну заробітну плату:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 76\,800 \cdot (1 + 0.2) = 92\,160 \text{ грн.}$$

Далі розраховується відрахування на соціальний внесок (СВІД), яке складає 22%:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 92\,160 \cdot 0{,}22 = 20275{,}2 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 20 000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 20\,000 = 5750 \text{ грн.},$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 25\,000 \cdot 0.08 = 2\,800 \text{ грн.},$$

де $К_Р$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 =$$
$$=1706.4 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706{,}4 \cdot 0{,}2 \cdot 0{,}9 \cdot 4{,}87 = 1495{,}83 \text{ грн грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_з$– коефіцієнтом зайнятості приладу;

$Ц_{ен}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 20000 \cdot 0{,}67 = 13400 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (4.17)$$

$$С_{ЕКС} = 92\,160 + 20275{,}2 + 5750 + 2800 + 1495{,}83 + 13400 = 135\,881.03 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 135\,881.03 / 622{,}57 = 218{,}258 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_м = C_{м\text{-}г} \cdot T, \quad (4.18)$$

I. $C_м = 218{,}258 \cdot 1328.64 = 289\,986.31$ грн.

II. $C_м = 218{,}258 \cdot 1345.52 = 293\,670.504$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_н = C_{зп}\, 0{,}67, \quad (4.19)$$

I. $C_н = 189\,806{,}32 \cdot 0{,}67 = 127\,170.2344$ грн.

II. $C_н = 192\,217{,}77 \cdot 0{,}67 = 128\,785.9059$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{пп} = C_{зп} + C_{від} + C_м + C_н, \quad (4.20)$$

I. $C_{пп} = 189\,806{,}32 + 41\,757.39 + 289\,986.31 + 127\,170.2344 = 648\,720{,}2544$ грн.

II. $C_{пп} = 192\,217{,}77 + 42\,287{,}9094 + 293\,670.504 + 128\,785.9059 = 656962.0893$ грн.

4.7 Вибір оптимального варіанту реалізації ПП

Розрахуємо коефіцієнт технічно-економіного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 4.68 / 648\,720{,}2544 = 7{,}2142 \cdot 10^{-6},$$

$$К_{ТЕР2} = 4.86 / 656962.0893 = 7{,}39769 \cdot 10^{-6}.$$

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР} = 7{,}39769 \cdot 10^{-6}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розробляється, можна зробити висновок, що з альтернатив, що

залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ = $7{,}39769 \cdot 10^{-6}$. Цей варіант реалізації програмного продукту має такі параметри:

– Мова програмування – Python;

– Використання необроблених даних даних;

– Використання готових модулів.

4.8 Висновки до розділу 4

В даній частині роботи було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту. Було визначено параметри, які характеризують програмний продукт. Проведено експертне оцінювання параметрів та аналіз якості варіантів реалізації функцій. Проведено економічний аналіз варіантів розробки – трудомісткість, витрати на заробітну плату та інші витрати.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

Отже, прогнозування відтоку користувачів є критично важливою задачею в телекомунікаційній сфері. Методи машинного навчання представляють собою потужний арсенал інструментів, що дозволяють аналізувати великі об’єми даних та робити досить точні передбачення, з якими можна працювати надалі, для формування стратегій збереження клієнтів та забезпечення стабільного розвитку бізнесу.

В даній дипломній роботі було здійснено побудову системи прогнозування відтоку на основі методів машинного навчання.

В першому розділі було проведено огляд предметної області, а саме телекомунікаційної галузі, зокрема, в Україні. Розглянуто проблему відтоку, його причини та вплив на бізнес. Проаналізовано сучасні інструменти управління відтоком.

В другому розділі було виконано постановку задачі. В даному випадку задача прогнозування відтоку аудиторії розглядалась як задача бінарної класифікації, в якій всі клієнти поділяються на два класи: тих, що залишаються з компанією і тих, що планують припинити взаємодію з нею. Було виконано огляд математичних основ обраних для дослідження методів машинного навчання, а також обґрунтовано вибір метрик якості для оцінювання прогностичних моделей.

В третьому розділі безпосередньо було описано процес прогнозування, що включав аналіз та попередню обробку даних, побудову моделей, налаштування гіперпараметрів для покращення їх якості та аналіз отриманих результатів. Усі моделі показали досить високу точність, а саме більш ніж 80% правильно класифікованих користувачів. Проте найкращі результати дали XGBClassifier (accurancy = 0.959567) та SVM (accurancy = 0.928246).

Якість моделей може бути покращена в майбутньому. Досить хороша прогностична здатність дозволяє інтегрувати їх у вже функціонуючі рішення та існуючі системи моніторингу відтоку тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про телекомунікації: Закон України від 18.11.2003 № 1280-IV. URL: https://zakon.rada.gov.ua/laws/show/1280-15#Text
2. В.М. Гранатуров, Н. Л.А. Захарченко, В.М. Орлов та ін. Економіка та менеджмент телекомунікацій. Одеса: Одеська національна академія зв'язку ім. О.С. Попова (ОНАЗ), 2015. 140 с.
3. Держкомстат України. URL: http://www.ukrstat.gov.ua.
4. Телекомунікації в Україні. URL: https://uk.wikipedia.org/wiki
5. ЗВІТ про роботу Національної комісії, що здійснює державне регулювання у сфері зв'язку та інформатизації НКРЗІ. 2021. URL: https://nkrzi.gov.ua/images/upload/662/10077/Dodatok_do_rishennia_27_01.04.2022.pdf.
6. Global Customer Experience KPMG. 2021. URL: https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2021/10/orchestrating-experiences.pdf.
7. Forrester. 2020. URL: https://www.forrester.com/bold.
8. Harvard Business School. URL: https://hbswk.hbs.edu/archive/the-economics-of-e-loyalty.
9. Gartner. URL: https://www.gartner.com/en.
10. Marketing Metrics. URL: https://www.issgovernance.com/market-intelligence/market-metrics/.
11. Forum Corporation. URL: https://www.achieveforum.com/resources-research.
12. B. Huang, M. Kehadi, B. Buckley. Customer churn prediction in telecoms, Expert System with Applications, 2012. 1425 с.
13. Національна комісія, що здійснює державне регулювання у сферах комунікацій. URL: https://nkrzi.gov.ua.

14. Оператор lifecell. Пдсумки фінансової та операційної діяльності за четвертий квартал 2022 року. 2022. URL: https://m.lifecell.ua/uploads/filelibrary/public/press_releases/UA_lifecell%20Q422_fin.pdf.
15. Cohort Analysis. Xpon Technologies Group. 2018. URL: https://xpon.ai/resources/how-to-use-cohort-analysis-to-measure-customer-lifetime-value/.
16. Y. Fareniuk, T. Zatonatska, O. Dluhopolsky. Customer Churn Prediction Model: A Case of the Telecommunication Market. 2022. URL: https://www.researchgate.net/publication/365454086_Customer_Churn_Prediction_Model_A_Case_of_the_Telecommunication_Market.
17. J. Vijaya, E. Sivasankar. Computing efficient features using rough set theory combined with ensemble classification techniques to improve the customer churn prediction in telecommunication sector. International Journal of Engineering & Technology. 2018.
18. R. Yahaya, O. Abisoye, S. Bashir. An enhanced bank customers churn prediction model using a hybrid genetic algorithm and k-means filter and artificial neural network. 2021.
19. Machine Learning Tutorial. 2023. URL: https://www.geeksforgeeks.org/machine-learning/
20. Недашківська Н. І. Вступ до інтелектуального аналізу даних. Київ: Інститут прикладного системного аналізу Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського, 2021. 35 с.
21. Класифікація в бізнес-аналітиці. Інформаційні системи та технології в управлінні. Запоріжжя: Запорізький національний технічний університет, 2014.
22. Логістична регресія. URL: https://uk.wikipedia.org/wiki/Логістична_регресія.

23. Hunt, Earl B., Janet Marin, Philip J. Stone Experiments in Induction. New York: Academic Press, 1966.
24. Hovland. Computer simulation of thinking, 1960.
25. Quinlan, J. Ross. Programs for Machine learning, 1993.
26. Quinlan, J. R. Induction of decision trees. Machine Learning, 1986.
27. Kelleher J.D., Namee B.M, D'Arcy A. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. The MIT Press, 2015. 624 p.
28. Decision trees in machine learning URL: https://www.jcchouinard.com/decision-trees-in-machine-learning/.
29. Aurelien Geron. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. Aurelien Geron., 2019. 1065p.
30. Naive bayes algorithm. URL: https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html.
31. Support vector. URL: https://en.wikipedia.org/wiki/Support_vector_machine.
32. Xgboost. URL: https://www.geeksforgeeks.org/xgboost/.
33. Cross validation. URL: https://scikit-learn.org/stable/modules/cross_validation.html.
34. Confusion matrix. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.
35. Metrics. URL: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/.
36. Python 3.11.3 documentation URL: https://docs.python.org/3/.
37. Dataset. URL: https://www.kaggle.com/datasets/blastchar/telco-customer-churn.

ДОДАТОК А

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import (
    MinMaxScaler, OneHotEncoder, LabelBinarizer,
StandardScaler
)
from sklearn.model_selection import (
    train_test_split, GridSearchCV, StratifiedKFold,
cross_val_score,KFold
)
from sklearn.metrics import (
    accuracy_score, confusion_matrix, f1_score,
    roc_curve, auc, classification_report,
    recall_score, precision_score, log_loss
)

telecom_customers =
pd.read_csv('telecom_customer_churn_final.csv')
# Data description
telecom_customers.shape
# Check the 5 first rows of the dataset.
telecom_customers.head()
telecom_customers.info()
# Describe columns with numerical values
telecom_customers.describe()
telecom_customers.corr(method='pearson')
# Number of customers that were retained and churned.
telecom_customers['Churn'].value_counts()
#Checking Data
categorial_features = ['gender', 'SeniorCitizen', 'Partner',
'Dependents', 'PhoneService', 'MultipleLines',
'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract',
'PaperlessBilling', 'PaymentMethod']
number_features = ['tenure','MonthlyCharges', 'TotalCharges']
for i in cat_features:
    print(i)
    print(churn[i].value_counts())
    print("----
telecom_customers.drop(columns='customerID', inplace=True)
telecom_customers
telecom_customers.drop(columns='ChurnReason', inplace=True)
telecom_customers.TotalCharges =
pd.to_numeric(telecom_customers.TotalCharges, errors='coerce')
```

```
# Data visualization
import plotly.graph_objects as go
from plotly.subplots import make_subplots
gender_labels = ['Male', 'Female']
churn_labels = ['No', 'Yes']
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'},
{'type':'domain'}]])
fig.add_trace(go.Pie(labels=gender_labels,
values=telecom_customers['gender'].value_counts(),
name="Gender"),
              1, 1)
fig.add_trace(go.Pie(labels=churn_labels,
values=telecom_customers['Churn'].value_counts(), name="Churn"),
              1, 2)
fig.update_traces(hole=.45, hoverinfo="label+percent+name",
textfont_size=16)
fig.update_layout(
    title_text="Gender and Churn Distributions",
    annotations=[dict(text='Gender', x=0.18, y=0.5,
font_size=25, showarrow=False),
                 dict(text='Churn', x=0.81, y=0.5, font_size=25,
showarrow=False)])
fig.data[0].marker.colors = ('lightblue', 'darkblue')
fig.data[1].marker.colors = ('lightgreen', 'green')
fig.show()


reason_counts =
telecom_customers['ChurnReason'].value_counts().tail(20)
fig, ax = plt.subplots(figsize=(7, 5))
colors = sns.color_palette('viridis', len(reason_counts))
reason_counts.plot(kind='barh', ax=ax, color=colors)
ax.set_title('Distribution of Customer Reasons for Churn')
ax.set_xlabel('Count')
ax.set_ylabel('Reason')
ax.invert_yaxis()
ax.tick_params(axis='y', pad=8)
ax.grid(False)
plt.show()

dem_col = ['gender', 'SeniorCitizen', 'Partner', 'Dependents']
acc_col = ['Contract', 'PaperlessBilling', 'PaymentMethod']
serv_col = ['PhoneService', 'MultipleLines', 'InternetService',
'OnlineSecurity',
            'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies']
fig, axes = plt.subplots(nrows=len(serv_col), ncols=1,
figsize=(7, 40))
colors = ['#256D85', '#FF4040']
for i, col in enumerate(serv_col):
  ax = axes[i]
  sns.countplot(data=telecom_customers, x=col, hue='Churn',
palette=colors, ax=ax)
```

```
  ax.set_title(f'Histogram for {col}')
  ax.set_xlabel(col)
  ax.set_ylabel('Count')
  ax.legend(['No Churn', 'Churn'])
  ax.grid(axis='y')
plt.tight_layout()
plt.show()

number_features = ['tenure','MonthlyCharges', 'TotalCharges']
fig, axes = plt.subplots(1, 3, sharey=True, figsize=(20, 4))
fig.suptitle('Custom Boxplots for Numeric Features', y=1,
size=25)
axes = axes.flatten()
for i, column in enumerate(number_features):
    sns.boxplot(data=telecom_customers[column], orient='h',
ax=axes[i], palette='Set3')
    axes[i].set_title(column + ', skewness: ' +
str(round(telecom_customers[column].skew(axis=0, skipna=True),
2)))
plt.tight_layout()

correlation =
df_copy.corr(method='pearson')['Churn'].drop('Churn')
correlation = correlation.sort_values()
plt.figure(figsize=(10, 6))
sns.barplot(x=correlation.index, y=correlation.values)
plt.xticks(rotation=90)
plt.xlabel('Features')
plt.ylabel('Correlation with Churn')
plt.show()

sns.set_context('poster', font_scale=0.6)
fig, ax = plt.subplots(1, 3, figsize=(20, 6))

ax1 = sns.histplot(x=telecom_customers['tenure'],
color='purple', hue=telecom_customers['Churn'], ax=ax[0],
bins=15, kde=True, palette='muted')
ax1.set(xlabel='Tenure', ylabel='Frequency')

ax2 = sns.histplot(x=telecom_customers['MonthlyCharges'],
color='blue', hue=telecom_customers['Churn'], ax=ax[1], bins=15,
kde=True, palette='pastel')
ax2.set(xlabel='Monthly Charges', ylabel='Frequency')

ax3 = sns.histplot(x=telecom_customers['TotalCharges'],
color='green', hue=telecom_customers['Churn'], ax=ax[2],
bins=15, kde=True, palette='dark')
ax3.set(xlabel='Total Charges', ylabel='Frequency')

plt.tight_layout()
plt.show()

# Data preparation
```

```
if telecom_customers.duplicated().sum() > 0:
    telecom_customers.drop_duplicates(inplace=True)
    print("Duplicates removed")
else:
    print("No duplicates found")

telecom_customers.isnull().values.any()
telecom_customers.isnull().sum().to_frame("counts")
telecom_customers['new_ver'] =
telecom_customers['MonthlyCharges'] *
telecom_customers['tenure']
telecom_customers['diff_in_charges'] =
telecom_customers['TotalCharges'] - telecom_customers['new_ver']
import plotly.express as px
fig = px.histogram(telecom_customers, x="diff_in_charges",color
= 'Contract')
fig.show()
telecom_customers['TotalCharges'] =
np.where(telecom_customers['TotalCharges'].isna() ==
True,telecom_customers['new_ver'],
telecom_customers['TotalCharges'])
telecom_customers =
telecom_customers.drop(['new_ver','diff_in_charges'], axis=1)

# creation of new features
SERVICES_ = ['PhoneService', 'MultipleLines', 'OnlineSecurity',
'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies']
telecom_customers["NumberOfServices"] =
(telecom_customers[SERVICES_] ==
"Yes").values.astype(int).sum(axis=-1) +
(telecom_customers["InternetService"] !=
"No").values.astype(int)
telecom_customers["TotalAvgChargesPerService"] =
telecom_customers["TotalCharges"] /
telecom_customers["NumberOfServices"]

# check unique values of each column
for column in telecom_customers.columns:
    print('In column {} there are {} unique values:
{}'.format(column, telecom_customers[column].nunique(),
telecom_customers[column].unique()))
telecom_customers['MultipleLines'] =
telecom_customers['MultipleLines'].replace('No phone service',
'No')
telecom_customers[['OnlineSecurity', 'OnlineBackup',
'DeviceProtection',
      'TechSupport', 'StreamingTV', 'StreamingMovies']] =
telecom_customers[['OnlineSecurity', 'OnlineBackup',
'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies']].replace('No internet
service', 'No')
```

```
telecom_customers['PaymentMethod'] =
telecom_customers['PaymentMethod'].str.replace(' (automatic)',
'', regex=False)
# check outliers
num_cols =
telecom_customers.select_dtypes(include=np.number).columns.tolis
t()
for col in num_cols:
    mean = telecom_customers[col].mean()
    std = telecom_customers[col].std()
    cutoff = std * 3
    lower, upper = mean - cutoff, mean + cutoff
    outliers = telecom_customers[(telecom_customers[col] <
lower) | (telecom_customers[col] > upper)]
    if outliers.empty:
        print(f'No outliers found in {col}')
    else:
        print(f'Outliers found in {col}:')
        print(outliers)
# label encoding
telecom_customers_transformed = telecom_customers.copy()
label_encoding_columns = ['gender', 'Partner', 'Dependents',
'PaperlessBilling', 'PhoneService', 'Churn']
for column in label_encoding_columns:
    if column == 'gender':
        telecom_customers_transformed[column] =
telecom_customers_transformed[column].map({'Female': 1, 'Male':
0})
    else:
        telecom_customers_transformed[column] =
telecom_customers_transformed[column].map({'Yes': 1, 'No': 0})
# one-hot encoding
one_hot_encoding_columns = ['MultipleLines', 'InternetService',
'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
                            'TechSupport', 'StreamingTV',
'StreamingMovies', 'Contract', 'PaymentMethod']
telecom_customers_transformed =
pd.get_dummies(telecom_customers_transformed, columns =
one_hot_encoding_columns)

# min-max normalization
min_max_columns = ['tenure', 'MonthlyCharges', 'TotalCharges',
'NumberOfServices', 'TotalAvgChargesPerService']
for column in min_max_columns:
        min_column = telecom_customers_transformed[column].min()
        max_column = telecom_customers_transformed[column].max()
        telecom_customers_transformed[column] =
(telecom_customers_transformed[column] - min_column) /
(max_column - min_column)


X = telecom_customers_transformed.drop(columns='Churn')
Y = telecom_customers_transformed.loc[:, 'Churn']
```

```
# split the data in training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.25, random_state=40, shuffle=True)
print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)
#oversampling
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 0)
x_sm, y_sm = sm.fit_resample(X_train, Y_train)
#undersampling
rus = RandomUnderSampler(random_state = 0)
x_rus, y_rus = rus.fit_resample(X_train, Y_train)
ax = plt.figure(figsize=(4,4)).add_subplot(111)
ax.set_title('Churn', color = 'black', size = 13)
sns.countplot(x=Y_train, palette='viridis', hue=Y_train)
ax.set_yscale('log')
ax.set_xlabel('Churn')
ax.set_ylabel('Number of Customers')
ax.legend(['Non-churn', 'Churn'])
ax.plot()
plt.tight_layout()
def quality_metrics( pred_test, pred_train, Y_test, Y_train):
    test_acc = accuracy_score(Y_test, pred_test)
    test_f1 = f1_score(Y_test, pred_test)
    test_prec = precision_score(Y_test, pred_test)
    test_rec = recall_score(Y_test, pred_test)
    test_falseNegRate = 1 - test_rec
    test_roc_auc = roc_auc_score(Y_test, pred_test)

    train_acc = accuracy_score(Y_train, pred_train)
    train_f1 = f1_score(Y_train, pred_train)
    train_prec = precision_score(Y_train, pred_train)
    train_rec = recall_score(Y_train, pred_train)
    train_falseNegRate = 1 - train_rec
    train_roc_auc = roc_auc_score(Y_train, pred_train)

    qua_metrics_df = pd.DataFrame({
        'Accuracy': [train_acc, test_acc],
        'F1 Score': [train_f1, test_f1],
        'Precision': [train_prec, test_prec],
        'Recall': [train_rec, test_rec]
    },
    index=['Train', 'Test'])

    return qua_metrics_df
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

def confusion_matrix_plot(cnf_matrix, title):
    cnf_matrix_pd = pd.DataFrame(data=cnf_matrix,
columns=['Predicted non-churn', 'Predicted churn'], index=['Non-
churn', 'Churn'])
```

```
    plt.figure(figsize=(6, 4))
    plt.title(title, fontsize=16)
    cmap = ListedColormap(['#f5f5f5', '#d9f0a3', '#a6d96a',
'#66bd63', '#1a9850'])
    sns.heatmap(cnf_matrix_pd, annot=True, fmt='d', cmap=cmap,
cbar=True, annot_kws={"fontsize":12})
    plt.xlabel('Predicted', fontsize=14)
    plt.ylabel('Actual', fontsize=14)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12, rotation=0)
    plt.show()
def plot_roc_curve(fpr, tpr, auc):
    plt.figure(figsize=(6, 4))
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr, tpr, linewidth=2, label="ROC Curve (AUC =
{:.3f})".format(auc), color='orange')
    plt.xticks(np.arange(0, 1, 0.05), rotation=90, fontsize=12)
    plt.yticks(fontsize=12)
    plt.xlabel('False Positive Rate', fontsize=15)
    plt.ylabel('True Positive Rate', fontsize=15)
    plt.legend(loc='lower right', fontsize=12)
    plt.title('Receiver Operating Characteristic (ROC) Curve',
fontsize=15)
    sns.despine()
    plt.show()
def Gini(x):
    v = 0
    for k, x_k in enumerate(x[:-1], 1):
        v += np.sum(np.abs(x_k - x[k:]))
    return v / (len(x)**2 * np.mean(x))
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix, roc_curve,
roc_auc_score, accuracy_score, f1_score, precision_score,
recall_score
import matplotlib.pyplot as plt
from xgboost import XGBClassifier
models = [
    ('Logistic Regression', LogisticRegression(max_iter=3000)),
    ('SVM', SVC(probability=True, kernel='linear'), ),
    ('Naive Bayes', GaussianNB()),
    ('Decision Tree', DecisionTreeClassifier()),
    ('XGBClassifier', XGBClassifier())
]
from sklearn.model_selection import GridSearchCV
param_grid = {
    'Logistic Regression': {
        'C': [0.1, 1, 10],
        'penalty': ['l1', 'l2'],
        'solver': ['liblinear', 'saga']
```

```
    },
    'SVM': {
        'C': [0.1, 1, 10],
        'gamma': [0.1, 0.01, 0.001],
        'kernel': ['linear', 'rbf', 'poly']
    },
    'Naive Bayes': {},

    'Decision Tree': {
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy'],
    'max_features': [None, 'sqrt', 'log2']
    },

    'XGBClassifier': {
    'max_depth': [3, 6, 9],
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 200, 300],
    'subsample': [0.8, 0.9, 1.0]
    }
}


for model_name, model_rus in models:
    model_rus.fit(x_rus, y_rus)
    pred_test_rus = model_rus.predict(X_test)
    pred_train_rus = model_rus.predict(x_rus)

    print('Quality_metrics for Model:', model_name)
    metrics_df = quality_metrics(pred_test_rus, pred_train_rus,
Y_test, y_rus)
    print(tabulate(metrics_df, headers='keys',
tablefmt='fancy_grid'))

    matrixForModel = confusion_matrix(Y_test, pred_test_rus)
    confusion_matrix_plot(matrixForModel, "Confusion matrix for
model")

    fpr, tpr, thresholds = roc_curve(Y_test,
model_rus.predict_proba(X_test)[:, 1])
    auc = roc_auc_score(Y_test,
model_rus.predict_proba(X_test)[:, 1])
    plot_roc_curve(fpr, tpr, auc)

    scores = cross_val_score(model_rus, x_rus, y_rus, cv=5,
scoring='recall')
    print('-----------------------------------------------------
----------------------')
    print('Cross-validated Recall scores are:', scores)
    print('-----------------------------------------------------
----------------------')
```

```
    print('Average Cross-validated Recall score:
{:.4f}'.format(scores.mean()))
    print('----------------------------------------------------
----------------------')
    print('Cross-validated Recall standard deviation:
{:.4f}'.format(scores.std()))

from tabulate import tabulate

best_models = {}
for model_name, model in models:
    grid_search = GridSearchCV(model, param_grid[model_name],
cv=5, scoring='recall')
    grid_search.fit(x_rus, y_rus)
    best_models[model_name] = grid_search.best_estimator_
    print('Best Parameters for {}: {}'.format(model_name,
grid_search.best_params_))
for model_name, model in best_models.items():
    model.fit(x_rus, y_rus)
    pred_test_best_param = model.predict(X_test)
    pred_train_best_param = model.predict(x_rus)

    print('Quality_metrics for Model:', model_name)
    metrics_df = quality_metrics(pred_test_best_param,
pred_train_best_param, Y_test, y_rus)
    print(tabulate(metrics_df, headers='keys',
tablefmt='fancy_grid'))

    matrixForModel = confusion_matrix(Y_test,
pred_test_best_param)
    confusion_matrix_plot(matrixForModel, "Confusion matrix for
model")

    if isinstance(model, SVC):
        model.probability = True

    fpr, tpr, thresholds = roc_curve(Y_test,
model.predict_proba(X_test)[:, 1])
    auc = roc_auc_score(Y_test, model.predict_proba(X_test)[:,
1])
    plot_roc_curve(fpr, tpr, auc)

    scores = cross_val_score(model, x_rus, y_rus, cv=5,
scoring='recall')
    print('Cross-validated Recall scores for {}:
{}'.format(model_name, scores))
    print('Average Cross-validated Recall score for {}:
{:.3f}'.format(model_name, scores.mean()))
    print('Cross-validated Recall standard deviation for {}:
{:.3f}'.format(model_name, scores.std()))
    print()
```

```
model_log_reg =
LogisticRegression(max_iter=3000,class_weight={0:0.4,1:0.6})
model_log_reg.fit(X_train,Y_train)
pred_train_model_log_reg = model_log_reg.predict(X_train)
pred_test_model_log_reg = model_log_reg.predict(X_test)
print('Quality_metrics for Model:', model_log_reg)
quality_metrics(pred_test_model_log_reg,
pred_train_model_log_reg, Y_test, Y_train)
matrixForModel = confusion_matrix(Y_test,
pred_test_model_log_reg)
confusion_matrix_plot(matrixForModel, "Confusion matrix for
model")

fpr, tpr, thresholds = roc_curve(Y_test,
model_log_reg.predict_proba(X_test)[:, 1])
auc = roc_auc_score(Y_test,
model_log_reg.predict_proba(X_test)[:, 1])
plot_roc_curve(fpr, tpr, auc)
test_gini = Gini(model_log_reg.predict_proba(X_test)[:, 1])
print('-------------------------------------------------------
--------------------------')
print('Gini:', test_gini)
scores = cross_val_score(model_log_reg, X_train, Y_train, cv=5,
scoring='recall')
print('-------------------------------------------------------
--------------------------')
print('Cross-validated Recall scores are:', scores)
print('-------------------------------------------------------
--------------------------')
print('Average Cross-validated Recall score:
{:.4f}'.format(scores.mean()))
print('-------------------------------------------------------
--------------------------')
print('Cross-validated Recall standard deviation:
{:.4f}'.format(scores.std()))
print('-------------------------------------------------------
--------------------------')
print("Actual values    :", Y_test.values[:30])
print("Predicted values :", pred_test_model_log_reg[:30])

model_log_reg_rus = LogisticRegression(solver='saga')
model_log_reg_rus.fit(x_rus,y_rus)
pred_train_model_log_reg_rus = model_log_reg_rus.predict(x_rus)
pred_test_model_log_reg_rus = model_log_reg_rus.predict(X_test)

print('Quality_metrics for Model:', model_log_reg_rus)
quality_metrics(pred_test_model_log_reg_rus,
pred_train_model_log_reg_rus, Y_test, y_rus)
matrixForModel = confusion_matrix(Y_test,
pred_test_model_log_reg_rus)
confusion_matrix_plot(matrixForModel, "Confusion matrix for
model")
```

```
fpr, tpr, thresholds = roc_curve(Y_test,
model_log_reg_rus.predict_proba(X_test)[:, 1])
auc = roc_auc_score(Y_test,
model_log_reg_rus.predict_proba(X_test)[:, 1])
plot_roc_curve(fpr, tpr, auc)

test_gini = Gini(model_log_reg.predict_proba(X_test)[:, 1])
print('-------------------------------------------------------------
---------------------------')
print('Gini:', test_gini)
scores = cross_val_score(model_log_reg_rus, x_rus, y_rus, cv=5,
scoring='recall')
print('-------------------------------------------------------------
---------------------------')
print('Cross-validated Recall scores are:', scores)
print('-------------------------------------------------------------
---------------------------')
print('Average Cross-validated Recall score:
{:.4f}'.format(scores.mean()))
print('-------------------------------------------------------------
---------------------------')
print('Cross-validated Recall standard deviation:
{:.4f}'.format(scores.std()))
print('-------------------------------------------------------------
---------------------------')
print("Actual values    :", Y_test.values[:30])
print("Predicted values :", pred_test_model_log_reg_rus[:30])


feat_importances = pd.DataFrame(coefficients.T,
index=x_rus.columns, columns=["Coefficient"])
feat_importances.sort_values(by='Coefficient', ascending=True,
inplace=True)
plt.figure(figsize=(10, 7))
plt.barh(feat_importances.index,
feat_importances["Coefficient"], color='green')
plt.xlabel("Coefficient", fontsize=12)
plt.ylabel("Features", fontsize=12)
plt.title("Feature Coefficients", fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.gca().invert_yaxis
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()

model_tree_rus = DecisionTreeClassifier()
model_tree_rus.fit(x_rus,y_rus)
pred_train_model_tree_rus = model_tree_rus.predict(x_rus)
pred_test_model_tree_rus = model_tree_rus.predict(X_test)

print('Quality_metrics for Model:', model_tree_rus)
quality_metrics(pred_test_model_tree_rus,
pred_train_model_tree_rus, Y_test, y_rus)
```

```
importances = model_tree_rus.feature_importances_
feat_importances = pd.DataFrame(importances,
index=x_rus.columns, columns=["Importance"])
feat_importances.sort_values(by='Importance', ascending=False,
inplace=True)

plt.figure(figsize=(9, 7))
plt.barh(feat_importances.index, feat_importances["Importance"],
color='green')
plt.xlabel("Importance", fontsize=12)
plt.ylabel("Features", fontsize=12)
plt.title("Feature Importance - Decision Tree", fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.gca().invert_yaxis()
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()

model_GaussianNB_rus = GaussianNB()
model_GaussianNB_rus.fit(x_rus,y_rus)
pred_train_model_GaussianNB_rus =
model_GaussianNB_rus.predict(x_rus)
pred_test_model_GaussianNB_rus =
model_GaussianNB_rus.predict(X_test)
print('Quality_metrics for Model:', model_GaussianNB_rus)
quality_metrics(pred_test_model_GaussianNB_rus,
pred_train_model_GaussianNB_rus, Y_test, y_rus)
rom sklearn.svm import SVC
model_SVC_rus = SVC(kernel='linear')
model_SVC_rus.fit(x_rus,y_rus)
pred_train_model_SVC_rus = model_SVC_rus.predict(x_rus)
pred_test_model_SVC_rus = model_SVC_rus.predict(X_test)

print('Quality_metrics for Model:', model_SVC_rus)
quality_metrics(pred_test_model_SVC_rus,
pred_train_model_SVC_rus, Y_test, y_rus)
```

ДОДАТОК Б

Дипломна робота на тему:

"Система інтелектуального аналізу даних для прогнозування відтоку аудиторії в телекомунікаційній галузі"

Виконала:
студентка групи КА-92
Гаврилко Дар'я Олегівна
Науковий керівник:
доц., к.т.н. Дідковська М.В.

Київ - 2023

1

Актуальність дослідження

Ринок телекомунікацій постійно зростає та розширюється, а конкуренція стала особливо жорсткою. Велика кількість компаній кожного дня змагаються за лояльність споживачів, які, в свою чергу, стають все більш вимогливими до якості отримуваних послуг і готові змінювати постачальника, якщо ті не задовольняють їх потреби. Компаніям критично важливо керувати відтоком аудиторії. Ефективним інструментом для розуміння та прогнозування відтоку є первина обробка даних та застосування машинного навчання.

Тож, в даній дипломні роботи буде проведено побудову прогностичних моделей машинного навчання, аналіз отриманих результатів та вибір найкращої моделі. Особливу увагу буде приділено підготовці даних та їх попередньому аналізу, з метою визначення патернів поведінки користувачів та виявлення важливих факторів, що потенційно впливають на відтік.

2

Об'єкт дослідження

Дані про користувачів телекомунікаційної компанії.

Предмет дослідження

Методи обробки статистичних даних та моделі машинного навчання, зокрема, логістична регресія, наївний баєсівський класифікатор, дерева рішень, метод опорних векторів та XGBoost.

Мета дипломної роботи

Побудова системи прогнозування відтоку аудиторії в телекомунікаційній галузі на основі методів машинного навчання.

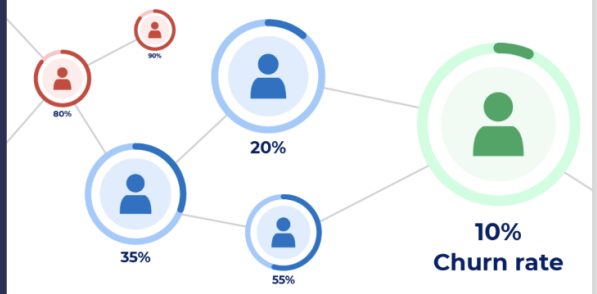
3

Аналіз предметної області

-- > Галузь телекомунікацій

-- > Проблема відтоку користувачів

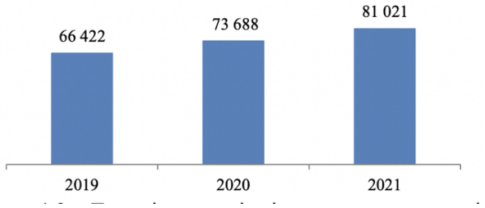
-- > Існуючі підходи до прогнозування відтоку



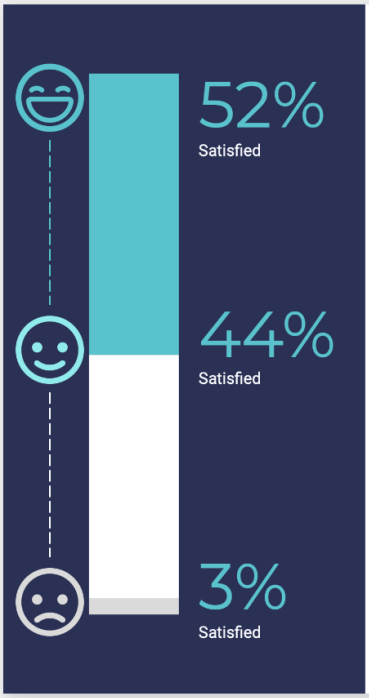
4

Галузь телекомунікацій

Сьогодні телекомунікації стали одним із ключових секторів економіки України, більше 20% загального обсягу послуг, що надаються в країні, реалізуються через діяльність операторів та провайдерів. У 2021 роуі доходи від надання послуг зв'язку склали 87 742 млн. грн.. За сотанні чотири роки інвестиції у галузь збільшилися на 15% та склали 13,2 млрд грн. Це свідчить про успішну роботу індустрії та доводить, що вона заслуговує на підтримку та увагу з боку регулюючих та законодавчих органів.



5



Проблема відтоку користувачів

Відтік клієнтів (customer churn) - це процес втрати споживачів товарів та/або послуг протягом певного періоду часу. Високий відтік вказує на те, що значна частина користувачів припиняє співпрацю з підприємством.

Типи відтоку:

-- > Природний
-- > Усвідомлений
-- > Прихований

Причини:

-- > Погана якість продуктів та послуг
-- > Неякісне обслуговування
-- > Велика кількість альтернативних компаній

6

Існуючі підходи до прогнозування відтоку

01

Машинне навчання

Полягає в застосування алго-ритмів машинного навчання, що дозволяє автоматизувати процес аналізу великих обсягів даних, що робить його більш ефективним та точним.

02

Когортний аналіз

Метод дослідження поведінки групи людей (когорти), які поділяють одну або декілька загальних характеристик.

03

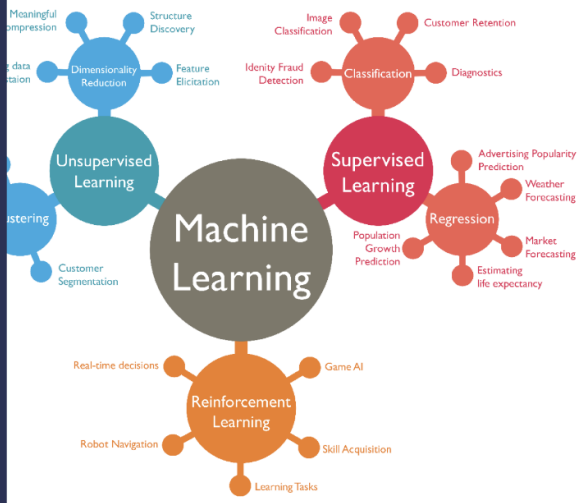
Аналіз тексту

Включає сентимент-аналіз, що допомагає визначати емоційне забарвлення тексту; веб-скрапінг; тематичний аналіз тощо.

7

Аналіз методів машинного навчання для практичної реалізації завдання класифікації

-- > Машинне навчання
-- > Постановка задачі класифікації
-- > Логістична регресія
-- > Дерева рішень
-- > Наївний Баєсівський класифікатор
-- > Метод опорних векторів
-- > Градієнтний бустинг
-- > Оцінювання якості моделей



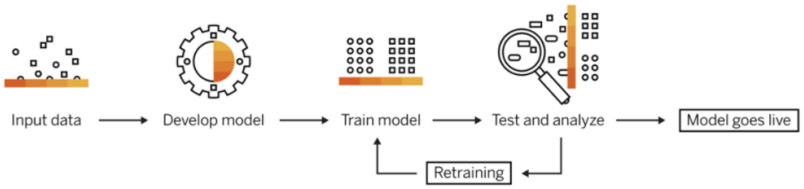
8

Машинне навчання

Машинне навчання - це напрям штучного інтелекту, що вивчає методи побудови алгоритмів, здатних навчатися. Алгоритми машинного навчання можна умовно розділити на три великі класи взалежності від досвіду та набору даних для навчання:

- навчання з учителем (supervised learning)
- навчання без учителя (unsupervised learning)
- навчання з частковим залученням учителя (semi-supervised)

Процес машинного навчання можна представити в наступних кроках:



9

Постановка задачі класифікації

Класифікація - це процес розбиття об'єктів заданого набору даних на групи (класи), що є апріорно відомими.

Дано:

X — Множина об'єктів (задана своїми ознаками, як точки n-вимірного простору)

y — Множина класів (у випадку бінарної класифікації {0,1})

Існує невідома цільова залежність - відображення y^*: X -> Y , значення якої відомі лише на елементах скінченної навчальної вибірки $X^m = \{(x_1,y_1),(x_2,y_2)...(x_m,y_m)\}$. Потрібно побудувати вирішальний алгоритм a: X -> Y , здатний класифікувати довільний об'єкт x Є X.

10

ЛОГІСТИЧНА РЕГРЕСІЯ

Усі регресійні моделі можуть бути записані в наступному вигляді:

$$y = F(x_1, x_2, \dots x_n)$$

У множинній лінійній регресії вважається, що залежна змінна є лінійною функцією від незалежних змінних, тобто:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$$

Подібно до лінійної регресійної моделі, логістична регресійна модель підраховує зважені суми вхідних ознак, але замість видачі результату безпосередньо, як це робить лінійна регресійна модель, вона видає логіт-перетворення результату.

11

ЛОГІСТИЧНА РЕГРЕСІЯ

Логіт-перетворення, представляє собою сигмоідальну (тобто S – подібної форми) функцію, що видає значення між 0 та 1. Вона визначена як показано в рівнянні та на рисунку нижче:

$$\hat{p} = \frac{1}{1 + \exp(-y)}$$

Після того як логістична регресійна модель оцінила ймовірність p^, що приклад x належить певному класу, вона може дати свій прогноз:

$$\hat{y} = \begin{cases} 0, \text{якщо } \hat{p} < 0.5 \\ 1, \text{якщо } \hat{p} \geq 0.5 \end{cases}$$

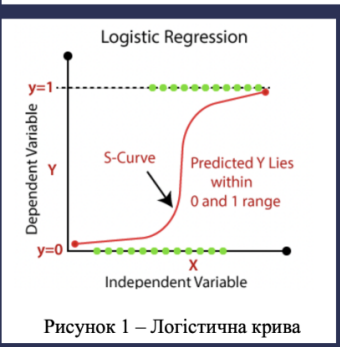
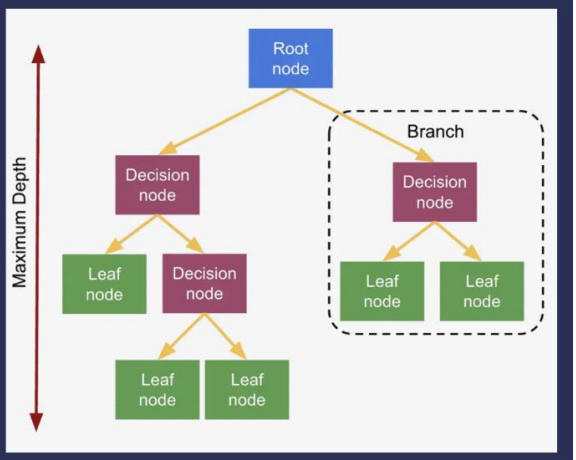


Рисунок 1 – Логістична крива

12

ДЕРЕВА РІШЕНЬ

Дерева рішень - це метод представлення правил у вигляді ієрархічної послідовної структури, де кожному об'єкту відповідає окремий вузол, що приймає рішення. У найпростішому випадку, множина прикладів, що потрапили до вузла, розбивається на дві підмножини, в одну з яких потрапляють приклади, що задовольняють правило, а в іншу - ті, що ні. Далі процес рекурсивно повторюється на кожній підмножині, доки не виконається умова зупинки алгоритму. Зрештою в останньому вузлі розбиття не проводять, і його оголошують листом.



13

НАЇВНИЙ БАЄСІВСЬКИЙ КЛАСИФІКАТОР

Це алгоритм навчання з учителем, принцип роботи якого базується на застосуванні теореми Баєса з «наївним» припущенням про незалежність ознак, які характеризують приклад з навчального набору даних.

Використовуючи Теорему Баєса, маємо:

$$p(C_k|x_1 \dots x_n) = \frac{p(C_k)p(x_1 \dots x_n|C_k)}{p(x_1 \dots x_n)}$$

Враховуючи вищезгадані припущення про незалежність ознак, рівняння вище набуває виду:

$$p(C_k|x_1 \dots x_n) = \frac{p(C_k)\prod_{i=1}^{n} p(x_i|C_k)}{p(x_1 \dots x_n)}$$

Відповідний класифікатор, представляє собою функцію, що назначає мітку класа C_k для деякого k наступним чином:

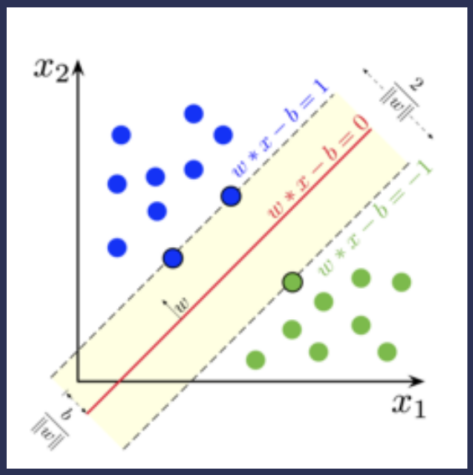
$$\hat{y} = \operatorname{argmax} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

14

МЕТОД ОПОРНИХ ВЕКТОРІВ

Суть методу опорних векторів (SVM) полягає в пошуку оптимальної гіперплощини, яка найкраще розділить дані на два класи на основі ознак прикладів.

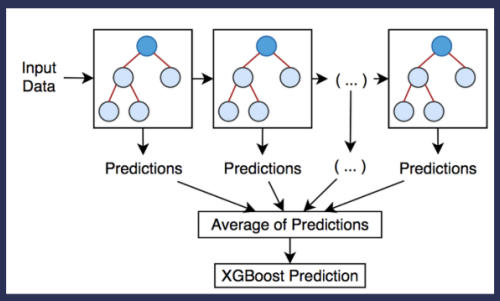
SVM використовує поняття опорних векторів - це навчальні приклади, що знаходяться найближче до границі розділення. Опорні вектори визначають положення та орієнтацію гіперплощини. Метою SVM є максимізувати відстань між гіперплощиною та найближчими до неї прикладами.



15

ГРАДІЄНТНИЙ БУСТИНГ

Це техніка машинного навчання, яка будує модель передбачення у вигляді ансамблю слабких моделей, якими зазвичай є дерева рішень. На кожній ітерації обчислюються відхилення прогнозів уже навченого ансамблю на навчальній вибірці. Нові дерева додаються до ансамблю доти, доки помилка зменшується, або доки не виконується одне з правил "ранньої зупинки".



Програмна реалізація виконувалась з допомогою бібліотеки XGBoost — одна з найпопулярніших та найефективніших реалізацій алгоритму градієнтного бустингу на деревах рішень.

16

Оцінювання якості моделей

Перехресна перевірка - статистичний метод, що дозволяє оцінити модель на незалежних даних, які не використовуються для навчання.

Матриця помилок – таблиця, яка дозволяє візуалізувати ефективність алгоритму класифікації шляхом порівняння прогнозованого значення цільової змінної з її фактичним значенням.

ROC-крива - дозволяє зрозуміти, як добре модель розрізняє класи, а не просто дивитися на точність передбачень при конкретному пороговому значенні.

Метрики якості:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

17

Етапи реалізації

01

Вибір моделей машинного навчання

Необхідно обрати моделі машинного навчання, що потенційно можуть бути ефективними для прогнозування відтоку аудиторії.

02

Збір, аналіз та візувалізвація даних

Цей етап включає дослідження зібраних даних, що дозволяє виявити кореляції, тренди та патерни, які можуть допомогти зрозуміти поведінку клієнтів.

03

Попередня обробка даних

Зібрані дані часто мають незадовільну якість, вони можуть містити шуми, пропущені чи помилкові значення, все це може потворювати прогнозування, тож важливим етапом є їх попередня обробка перш ніж навчати на них моделі.

04

Побудова моделей машинного навчання

На цьому етапі проводиться побудова прогностичних моделей машинного навчання.

05

Оцінка моделей та вибір найкращої

Останній етап включає оцінку якості побудованих моделей прогнозування, що дозволяє визначити ефективність моделей і обрати найкращу.

18

01

Вибір моделей машинного навчання

Обрані моделі машинного навчання:
- логістична регресія
- дерева рішень
- наївний Баєсівський класифікатор
- метод опорних векторів
- XGBoost

02

Збір, аналіз та візувалізвація даних

Датасет містить інформацію про 7043 абонентів, кожен з яких характеризується 23-ма ознаками. Цільовою змінною є атрибут Churn.

	customerID	gender	SeniorCitizen	ZIPcode	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	TechSupport
0	7590-VHVEG	Female	0	90003	Yes	No	1	No	No phone service	DSL	...	No
1	5575-GNVDE	Male	0	90005	No	No	34	Yes	No	DSL	...	No
2	3668-QPYBK	Male	0	90006	No	No	2	Yes	No	DSL	...	No
3	7795-CFOCW	Male	0	90010	No	No	45	No	No phone service	DSL	...	Yes
4	9237-HQITU	Female	0	90015	No	No	2	Yes	No	Fiber optic	...	No

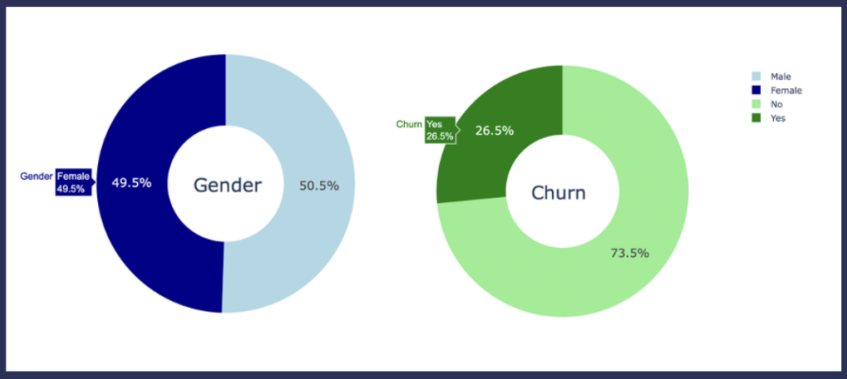
5 rows × 23 columns

19

02

Збір, аналіз та візувалізвація даних

Як бачимо з кругової діаграми дані є незбалансованими, це вказує на нерівномірний розподіл класів (у даному випадку, користувачів, які відтікають і ті що не мають таких намірів). 73.5% аудиторії компанії продовжує активну взаємодію з нею і тільки 26.5% має наміри розірвати контракт.

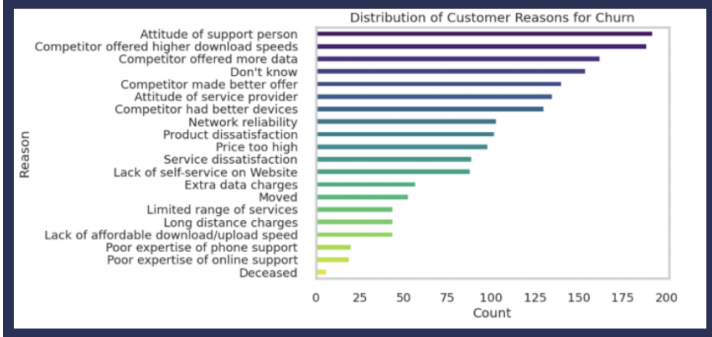


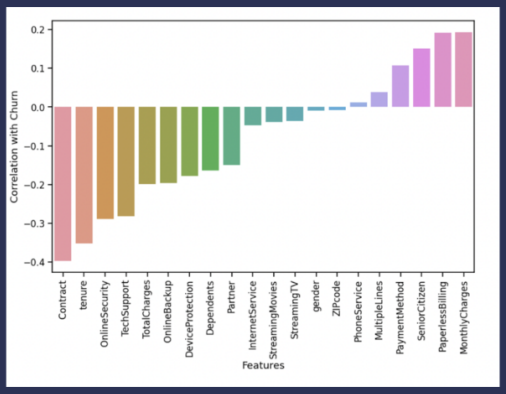
20

02

Збір, аналіз та візувалізвація даних

- Датасет, на основі якого проводиться дослідження, містить таку ознаку як ChurnReason, саме вона і допоможе проаналізувати причини, які спонукають споживачів шукати нових постачальників телекомунікаційних послуг.
- Також розглянемо кореляцію Пірсона між вхідними ознаками і цільовою змінною (відтоком).



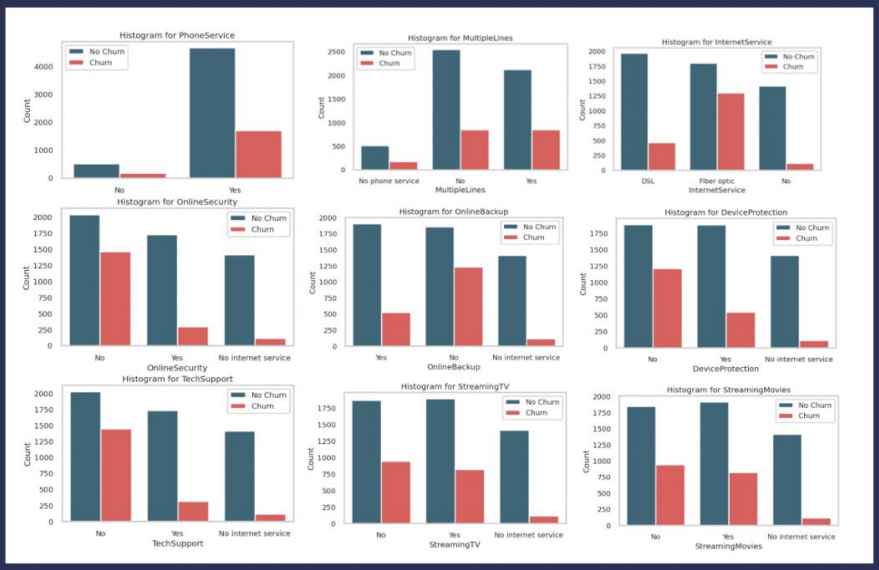


21

02

Збір, аналіз та візувалізвація даних

Розподіл цільової змінної за послугами, якими користуються абоненти:



22

03

Попередня обробка даних

1) Датасет було перевірено на наявність дуплікатів, після чого їх було видалено

2) Були видалені ознаки, що мають низьку кореляцію з цільовою змінною та не несуть корисної для прогнозування інформації (ZIPcode, CustimerID)

3) Було здійснено перевірку на наявність викидів, жодних виявлено не було

4) Датасет було перевірено на наявність пропущених значень. За ознакою TotalCharges було виявлено 11 таких значень. Видалення значень не проводилось, натомість було виявлено, що ефективним буде заповнення їх за формулою Tenure*MonthlyCharges.

5) З метою отримання ще більше інофрмації про датасет було створено додаткові ознаки (NumberServices та TotalAvgChargesPerService)

23

03

Попередня обробка даних

6) Було проведено перекодування категоріальних ознак набору даних з застосуванням Label Encoding та One-Hot Encoding.

7) Числові змінні були нормалізовані з застосуванням Min-Max Normalization:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

8) Було отримано готовий до подальшого моделювання набір даних:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	Churn	...
0	1	0	1	0	0.013889	0	1	0.115423	0.003437	0	...
1	0	0	0	0	0.472222	1	0	0.385075	0.217564	0	...
2	0	0	0	0	0.027778	1	1	0.354229	0.012453	1	...
3	0	0	0	0	0.625000	0	0	0.239303	0.211951	0	...
4	1	0	0	0	0.027778	1	1	0.521891	0.017462	1	...
...	...	...	...	...	...	...	...	...	...	...	...
7038	0	0	1	1	0.333333	1	1	0.662189	0.229194	0	...
7039	1	0	1	1	1.000000	1	1	0.845274	0.847792	0	...

24

03
Попередня обробка даних

9) Перед проведенням моделювання набір даних було поділено. Використовуючи метод train_test_split з бібліотеки scikit-learn, дані були розділені на навчальну та тестову вибірки(25% (1756 абонентів) вхідних даних будуть використані для тестування, а 75% (5265 абонентів)).

10) Для балансування даних були використані техніки undersampling та undersampling.

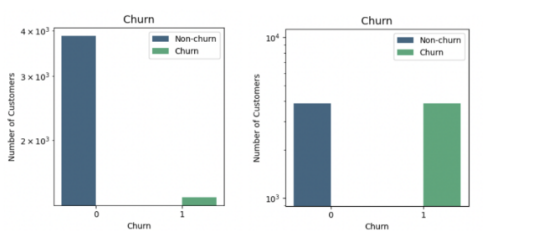


Рисунок 1 – Результат балансування класів технікою oversampling

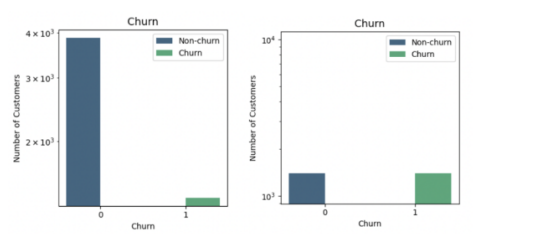


Рисунок 2 – Результат балансування класів технікою undersampling

04
Побудова моделей машинного навчання

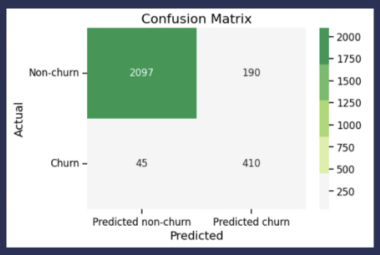
Аналіз результатів для логістичної регресії

В процесі підбору гіперпараметрів найкращі результати були отримані при комбінації: 'C': 1, 'penalty': l2, 'solver': 'liblinear'.

Аналіз результатів для дерев рішень

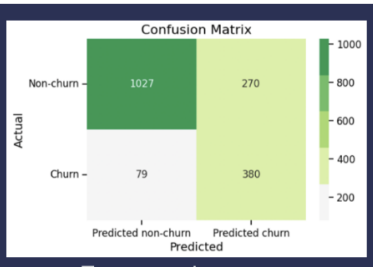
В процесі підбору гіперпараметрів найкращі результати були отримані при комбінації: 'criterion': entopy, 'max_depth': 5, 'max_features': 'log 2', 'min_samples_leaf': 4, 'min_samples_split': 5

	Accuracy	F1 Score	Precision	Recall
Train	0.922851	0.801887	0.702479	0.934066
Test	0.903720	0.777251	0.683333	0.901099



Логістична регресія

	Accuracy	F1 Score	Precision	Recall
Train	0.805180	0.695423	0.601218	0.824635
Test	0.792793	0.673162	0.584615	0.793319

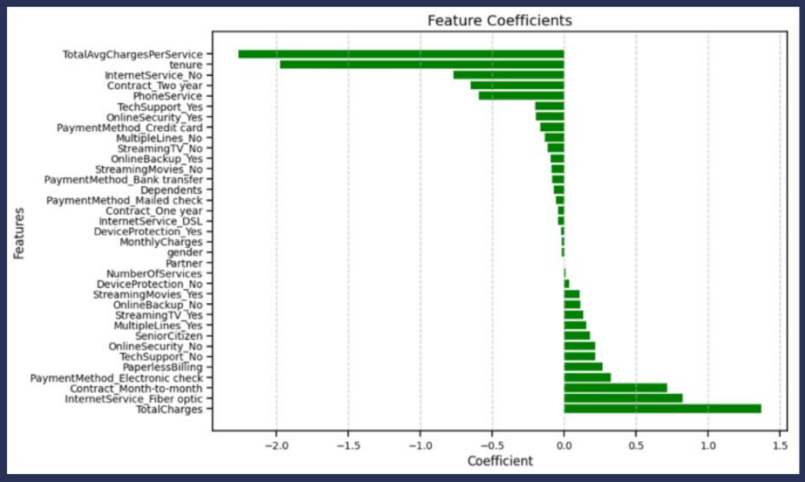


Дерева рішень

04

Побудова моделей машинного навчання

За допомогою атрибуту .coef_ логістична регресія дозволяє оцінити важливість ознак при прогнозуванні. Кожна ознака має свій коефіцієнт, що відображає, наскільки значущою вона є для прогнозування цільової змінної.

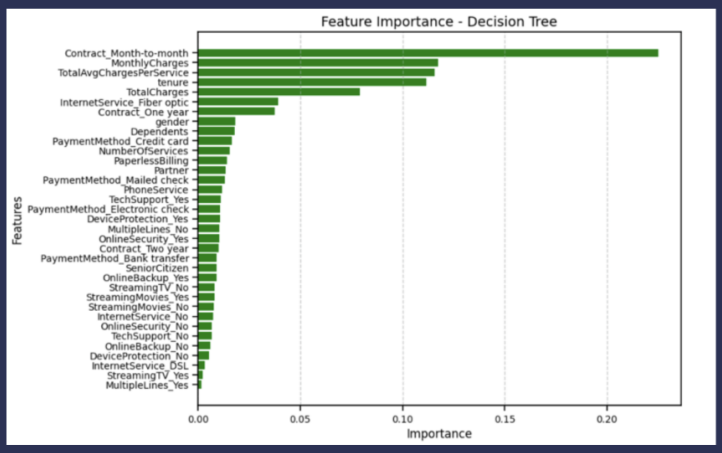


27

04

Побудова моделей машинного навчання

Розглянемо також атрибут feature_importances_, який є доступним для моделі, що розглядається, і вказує наскільки сильно кожна ознака впливає на прийняття рішень моделлю під час прогнозування.



28

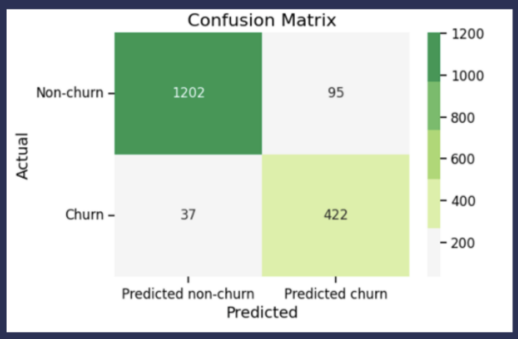
04

Побудова моделей машинного навчання

Аналіз результатів для наївного Баєсівського класифікатора

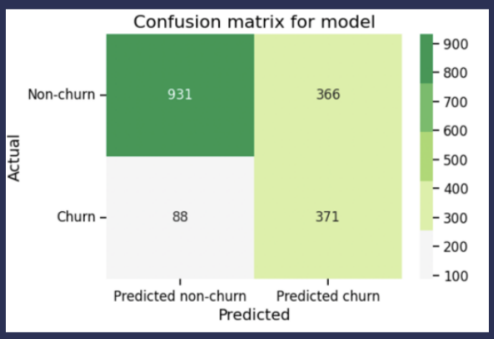
Збалансовані дані

	Accuracy	F1 Score	Precision	Recall
Train	0.942483	0.894901	0.856574	0.936819
Test	0.901481	0.864754	0.816248	0.919390



Незбалансовані дані

	Accuracy	F1 Score	Precision	Recall
Train	0.756795	0.765517	0.739015	0.793991
Test	0.741458	0.620401	0.503392	0.808279



29

04

Побудова моделей машинного навчання

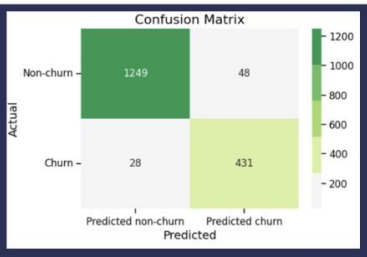
Аналіз результатів для методу опорних векторів

В процесі підбору гіперпараметрів найкращі результати були отримані при комбінації: 'C': 10, 'gamma': 0.001, 'kernel': 'rbf'.

Аналіз результатів для XGBoost

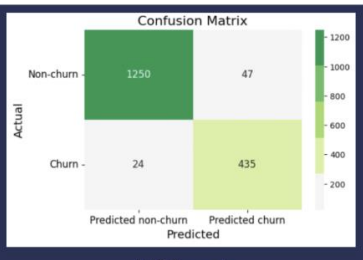
В процесі підбору гіперпараметрів найкращі результатами дала наступна комбінація: 'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 200, 'subsample': 0.9.

	Accuracy	F1 Score	Precision	Recall
Train	0.962984	0.930481	0.913866	0.947712
Test	0.928246	0.918977	0.899791	0.938998



Метод опорних векторів

	Accuracy	F1 Score	Precision	Recall
Train	0.965831	0.936170	0.914761	0.958606
Test	0.959567	0.924548	0.902490	0.947712



XGBoost

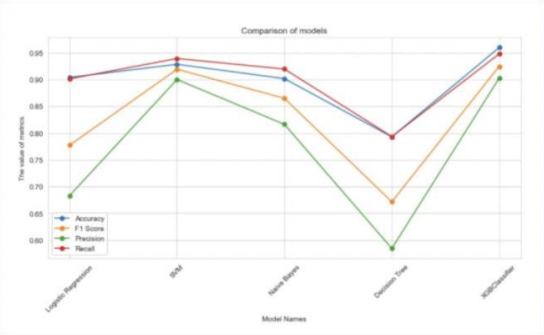
30

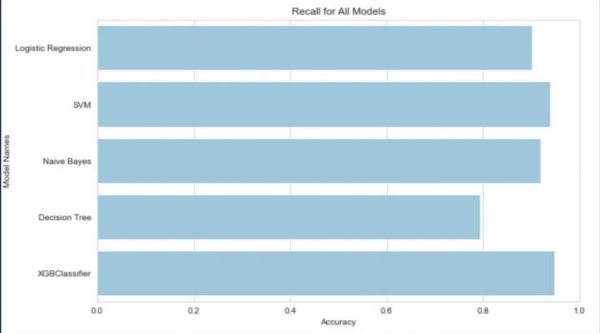
05

Оцінка моделей та вибір найкращої

Отже, маємо:

Model	Accurancy	F1 Score	Precision	Recall
Logistic Regression	0.903720	0.777251	0.68333	0.901099
SVM	0.928246	0.918977	0.899791	0.938998
Naive Bayes	0.901481	0.864754	0.816248	0.919390
Decision Tree	0.792793	0.67162	0.584615	0.793319
XGBClassifier	0.959567	0.924548	0.902490	0.947712





31

Висновки

Тож, було виконано аналіз та обробку вхідних даних, побудову обраних для класифікації методів машинного навчання та проведено їх оцінку з допомогою визначених в роботі метрик якості. Усі моделі показали досить високу точність, а саме більш ніж 80% правильно класифікованих користувачів. Проте найкращі результати дали XGBClassifier (accurancy = 0.959567) та SVM (accurancy = 0.928246), що дає можливіть використовувати їх для прогнозування відтоку користувачів та прийняття відповідних стратегічних рішенньь для збереження аудиторії.

Якість моделей може бути покращена в майбутньому. Досить хороша прогностична здатність дозволяє інтегрувати їх у вже функціонуючі рішення та існуючі системи моніторингу відтоку тощо.

32

ДЯКУЮ ЗА УВАГУ!