НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 519.245+004.896

До захисту допущено
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2024 р.

**Магістерська дисертація**
на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз фінансового ринку»
зі спеціальності 124 «Системний аналіз»
на тему: «Розв’язання задачі заповнення пропусків даних альтернативними методами при створенні прогнозних моделей»

Виконав:
Студент 2 курсу, групи КА-22мп
Попов Андрій Юрійович _____

Науковий керівник:
Професор кафедри ММСА, д.ф.-м.н., проф.
Макаренко Олександр Сергійович _____

Рецензент:
Професор кафедри ШІ, д.т.н, проф.
Данилов Валерій Якович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань
Студент (*підпис*): _____

Київ - 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність — 124 «Системний аналіз»
Освітньо-професійною програмою «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___» _____ 2024 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Попову Андрію Юрійовичу
(прізвище ім'я по батькові)

1. Тема дисертації: «Розв'язання задачі заповнення пропусків даних альтернативними методами при створенні прогнозних моделей»,
науковий керівник дисертації Макаренко О.С., д.ф.-м.н. професор,
(прізвище ім'я по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «08» листопада 2023 р. № 5200-с.
2. Строк подання студентом дисертації _____

3. Об'єкт дослідження: методи обробки пропущених даних у задачах аналізу даних.

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за ОПП): методи заповнення та їх використання у задачах попередньої обробки даних при навчанні моделей прогнозування.

5. Перелік завдань, які потрібно розробити:
- здійснити огляд технічної літератури за темою дисертації;
- дослідити актуальність обраної теми;
- дослідити теорію пропущених даних;
- визначитись із даними для моделювання;
- дослідити існуючі математичні методи та моделі для обробки пропущених даних;

- розробити програмний продукт, для проведення дослідження;
- провести експериментальне моделювання для дослідження ефективності обраних методів заповнення у поєднанні з обраними моделями прогнозування, провести порівняльну роботу;
- розробити стартап-проект виведення результатів дослідження на ринок;
- розробити концептуальні висновки за результатами дослідження.

6. Перелік графічного (ілюстративного) матеріалу: аналіз обраного набору даних, результати роботи програмного продукту на прикладі обраного набору даних, таблиці у розділі стартап-проекту.

7. Орієнтовний перелік публікацій:
- Попов А.Ю., Макаренко О.С., Бідюк П.І. Розв'язанні задачі заповнення пропусків даних альтернативними методами при створенні прогнозних моделей, II Всеукраїнська науково-практична конференція «Системні науки та інформатика», 4–8 грудня 2023 р., Київ : КПІ ім. Ігоря Сікорського, 2023. С. 201-206.

8. Консультанти розділів дисертації

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |

9. Дата видачі завдання: 01.09.2023

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Строк виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Затвердження теми магістерської дисертації та огляд технічної літератури. | 01.09.2023-15.09.2023 | Виконано. |
| 2 | Концептуальний вступ дисертації. Формулювання об'єкта, предмета, цілі, завдань, актуальності, практичної значущості результатів. | 15.09.2023-22.09.2023 | Виконано. |

| | | | |
|---|---|---|---|
| 3 | Визначення наборів даних для моделювання, їх аналіз та упорядкування. | 22.09.2023-29.09.2023 | Виконано. |
| 4 | Перший розділ. Дослідження загальної проблематики пропущених даних. | 29.09.2023-06.10.2023 | Виконано. |
| 5 | Другий розділ. Аналіз існуючих методів та підходів для обробки пропущених даних. | 06.10.2023-20.10.2023 | Виконано. |
| 6 | Розробка початкового варіанту програмного забезпечення. | 20.10.2023-27.10.2023 | Виконано. |
| 7 | Третій розділ. Експериментальне моделювання та аналіз отриманих результатів. | 27.10.2023-03.11.2023 | Виконано. |
| 8 | Четвертий розділ. Розробка стартап-проекту. | 03.11.2023-10.11.2023 | Виконано. |
| 9 | Висновки по роботі і перспективи подальших досліджень. | 10.11.2023- 20.11.2023 | Виконано. |
| 10 | Оформлення магістерської дисертації | 20.11.2023-31.12.2023 | Виконано. |

Студент _____ Андрій ПОПОВ

Науковий керівник дисертації_____ Олександр МАКАРЕНКО

# РЕФЕРАТ

Магістерська дисертація: 150 с., 8 рис., 36 табл., 1 додаток, 25 джерел.

Тема магістерської дисертації: «Розв’язання задачі заповнення пропусків даних альтернативними методами при створенні прогнозних моделей».

Мета роботи – дослідження впливу методів заповнення пропущених даних на результати навчання моделей прогнозування, розробка програмного забезпечення для проведення дослідження.

Об’єкт дослідження: методи обробки пропущених даних у задачах аналізу даних.

Предмет дослідження: методи заповнення та їх використання у задачах попередньої обробки даних при навчанні моделей прогнозування.

Отримані результати – побудоване спеціалізоване програмне забезпечення мовою Python, що надає змогу досліджувати результати роботи методів обробки пропущених даних на наборах даних з різними типами пропусків, а також вплив роботи методів на результативність моделей прогнозування, що використовують зазначені дані для навчання.

ПРОПУЩЕНІ ДАНІ, МЕТОДИ ЗАПОВНЕННЯ, МЕХАНІЗМИ ПРОПУЩЕНИХ ДАНИХ, ПРОГНОЗНІ МОДЕЛІ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ.

# ABSTRACT

Master's thesis: 150 pp., 8 fig., 36 tabl., 1 appendix, 25 sources.

Research topic: Alternative missing data imputation methods in predictive model creation.

Research goal – research of the influence of missing data imputation methods on the quality of predictive models, developing software to conduct the research.

Object of the study: missing data handling in data analysis.

Subject of the study: imputation methods and their usage in data preprocessing for predictive model fitting.

Obtained results – we developed software using Python programming language that allows researching results of imputation method usage on datasets with different missing data configurations as well as the impact the methods have on the predictive quality of forecasting models.

MISSING DATA, IMPUTATION METHODS, MISSING DATA MECHANISMS, PREDICTIVE MODELS, MACHINE LEARNING, DATA ANALYSIS.

ЗМІСТ

# ВСТУП

На сьогоднішній день будь-яка задача прогнозування пов’язана з необхідністю обробки величезних масивів даних. Ця величезна кількість інформації має неймовірний потенціал для відкриття нових знань і розкриття широкого спектру можливостей у різних сферах, включаючи медицину, фінанси, транспорт, та багато інших. Проте, використання цих даних у машинному навчанні пов'язане з реальними викликами, що полягають у зіставленні, очищенні та попередній обробці цих даних.

Одним із ключових аспектів підготовки даних для побудови прогнозних моделей є обробка пропущених даних. Відсутність інформації у даних може серйозно вплинути на якість та надійність моделі, спотворити статистичні параметри та призвести до неправильних висновків. У роботі проведено опис використовуваних методів обробки та заповнення пропущених даних, та аналіз їх впливу на деякі прогнозні моделі машинного навчання.

У першому розділі проводиться огляд загальної проблематики обробки пропущених даних.

У другому розділі розглядаються найпоширеніші методи роботи з пропущеними даними.

У третьому розділі описано розроблене програмне забезпечення для проведення дослідження на прикладі задачі прогнозування відтоку клієнтів, надається аналіз отриманих результатів.

Четвертий розділ присвячено розробці стартап-проекту на основі розробленого програмного забезпечення.

# РОЗДІЛ 1 АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ОСОБЛИВОСТЕЙ ПОПЕРЕДНЬОЇ ОБРОБКИ ДАНИХ

## 1.1 Актуальність дослідження

Задачі прогнозування з використанням алгоритмів машинного навчання на сьогоднішній день є надзвичайно важливими. З ростом обсягу даних та розвитком технологій аналізу, здатність передбачати майбутні події та тенденції стала ключовим аспектом у різних галузях. У бізнесі прогнозування допомагає приймати обґрунтовані рішення щодо запасів, виробництва та маркетингу, щоб підтримати конкурентоспроможність і оптимізувати витрати. У сфері фінансів прогнози грають критичну роль у прийнятті інвестиційних рішень та управлінні ризиками. В медицині прогнозування може бути використано для діагностики, передбачення поширення захворювань і розробки нових методів лікування. В науці й технологіях, прогнозування дозволяє передбачити кліматичні зміни, допомогти з розвитком нових матеріалів і технологій, тощо. У сфері транспорту прогнози допомагають оптимізувати маршрути та спланувати обслуговування. У випадках більших масштабів задач прогнозування також може використовуватись при довгостроковому плануванні. Машинне навчання в цьому контексті стає незамінним інструментом, який дозволяє аналізувати величезні обсяги даних та створювати моделі, що можуть прогнозувати різні явища. Використовуючи методи штучного інтелекту, машинне навчання допомагає покращити якість прогнозів та робити їх більш точними.

Машинне навчання вивчає алгоритми, що здатні покращувати власні характеристики прогнозування за допомогою накопичення досвіду, тому важливість якісних даних в контексті навчання та роботи алгоритмів машинного навчання не може бути переоцінена. Якість даних є фундаментальним фактором,

який визначає ефективність та надійність будь-якої моделі машинного навчання. Навіть найпотужніший алгоритм не зможе забезпечити точні та корисні результати, якщо дані, на яких він навчається, є неточними, неповними або забрудненими помилками. Якісні дані дозволяють алгоритмам вірно вивчати закономірності і зв'язки в даних, що відображають реальний світ. Це важливо для точних прогнозів, задач класифікації та прийняття рішень, призначених для оптимізації бізнес-процесів, медицини, фінансів і багатьох інших галузей. Низька якість даних може призвести до навчання алгоритмів на помилкових зразках, що в свою чергу призведе до невірних результатів і відповідно некоректних дій. Для забезпечення якості даних важливо проводити їх очищення, нормалізацію та перевірку на відповідність задачі.

Одним з важливих етапів попередньої обробки даних є перевірка на наявність пропущених даних та їх обробка. У реальних наборах даних часто зустрічаються пропущені значення, що можуть виникнути з різних причин, таких як технічні помилки, людський фактор, природу дослідження, відсутність даних у певний момент часу і багато інших. Навчання алгоритмів на даних з пропущеними значеннями може призвести до серйозних спотворень та невірних результатів, що вплине на роботу моделі. Обробка пропущених даних має на меті заповнення цих прогалин точними значеннями або їх видалення, якщо відповідна інформація недоступна. Цей процес може включати методи заповнення, яких існує велике різноманіття: від найпростіших, що лише заповнюють пропуски фіксованим значенням, до більш складних, що використовують статистику та залежності між іншими ознаками для визначення значення, яким виконується заповнення. Правильна обробка пропущених даних перед навчанням моделей гарантує, що моделі машинного навчання будуть працювати надійно та надавати корисні результати, сприяючи досягненню успіху у відповідних задачах.

При цьому важливо пам'ятати, що некоректна обробка пропущених даних може призвести до перенавчання моделі, підвищення шуму та загрози її надійності. Якість роботи методів заповнення може різнитися в залежності від області застосування. Через це при виборі методів обробки пропущених даних важливо досліджувати наявні дані, механізм утворення пропусків, враховувати контекст та специфіку задачі, оскільки саме ці фактори є визначальними при виборі методів заповнення.

## 1.2 Формалізація постановки задачі заповнення

Розглянемо набір даних D – це матриця, де кожний рядок є спостереженням (записом), а кожен стовпчик – ознакою (змінною). Задача заповнення даних полягає у пошуку функції $F(x)$, що заповнюватиме пропущені значення на основі наявних даних. Така функція може залежати як лише від значень у стовпчику з пропуском, так і від всіх даних, що є в таблиці. Існує широке різноманіття методів заповнення пропусків, що істотно різняться складністю та якістю. Перш ніж переходити до їх розгляду варто формалізувати класифікацію схем та механізмів утворення пропусків. Класифікація механізмів була введена Дональдом Б. Рубіном у роботі “Inference and Missing Data” (1976). Важливо розрізняти поняття механізму пропущених даних та схеми пропущених даних. Схема – це загальне поняття, що використовується для опису місць з пропусками в даних, але не описує причину утворення цих пропусків. Механізм – це термін, що покликаний в загальному вигляді описати зв’язки між пропусками та наявними даними. Розглядається шість типів схем пропущених даних.

1. Одновимірні – випадок коли пропуски присутні лише в одній спостережуваній змінній. Трапляється у дослідженнях, де присутня одна

цільова змінна, а всі інші знаходяться під контролем дослідника і пропусків не мають.

2. Unit Nonresponse Pattern – випадок коли частина змінних мають повні дані, а частина мають пропуски. Часто трапляється у аналізі опитувань коли учасники відмовляються відповідати на певні питання. Також може бути результатом планованих пропусках при організації опитувань.
3. Монотонні – випадок, що зазвичай трапляється у довготривалих дослідженнях, в яких деякі учасники можуть вибувати до закінчення експерименту.
4. Стандартний – випадок випадкового розташування пропусків по змінних в даних. Випадковість може бути як справжньою, так і оманливою коли між пропусками та даними є зв’язок.
5. Сплановані пропуски – випадок, що трапляється при організації опитувань коли за планом проведення не кожний набір питань, що надається учаснику, є повним.
6. Латентні змінні – випадок з повністю відсутніми даними по латентній змінній.

У даній роботі досліджується стандартна схема пропущених даних. Розглянемо основні механізми утворення пропусків. Загалом їх існує три типи.

- MAR (англ. Missing at random) – припущення, що передбачає, що ймовірність пропущених даних по змінній $Y$ залежить від значення по одній або декількох інших ознак, але не залежить від значення самої змінної $Y$. Іншими словами, немає зв’язку між ймовірністю відсутності даних у $Y$ та значеннями $Y$ після вилучення інших змінних. Термін "випадкова відсутність" дещо вводить в оману, оскільки він підказує, що дані відсутні в випадковому порядку, схожому на підкидання монети. Однак MAR фактично означає, що існує систематичний зв'язок між однією

або кількома вимірюваними змінними та ймовірністю відсутніх даних. MAR є основним припущенням про механізм пропусків, виконання якого є необхідним для використання складних методів заповнення, таких як метод максимальної правдоподібності та метод множинного заповнення. Але на практиці чітко встановити, що це припущення виконується, є неможливим, оскільки зазвичай встановити, що існує зв’язок між пропуском $Y$ та значенням $Y$, можна лише наперед знаючи це саме значення $Y$.

- MCAR (англ. Missing completely at random) – ймовірність пропущених даних по змінній $Y$ не залежить від значення по жодній іншій ознаці та не залежить від значення $Y$. Інакше кажучи, спостережувані дані є простою випадковою вибіркою записів, що можна було б аналізувати, якби дані були повними. MCAR є більш обмеженою умовою, ніж MAR, оскільки цей тип передбачає, що відсутність даних повністю не пов'язана з інформацією в наборі даних. Підтвердити на практиці, що пропуски відносяться до MCAR типу є можливим – за визначенням цей тип вказує на абсолютно випадкові пропуски, що говорить про те, що статистичні параметри даних з пропусками повинні збігатися в межах похибки з статистичними параметрами даних без пропусків. Якщо помітна різниця по якійсь змінній – це гарний індикатор того, що ймовірність пропусків залежить від тієї змінної, а отже, наявні пропуски не належать до MCAR типу.
- MNAR (англ. Missing not at random) – найслабше припущення, що передбачає, що ймовірність пропущених даних по змінній $Y$ залежить в тому числі і від значення самої змінної $Y$. Як у випадку MAR чітко встановити, що пропуски належать саме до типу MNAR є неможливим, оскільки для цього необхідно знати пропущене значення $Y$.

Для більш формального визначення механізмів пропущених даних позначимо спостережувані дані як $Y_{\text{с}}$, а пропущені як $Y_{\text{п}}$. Основна ідея в теорії Рубіна полягає у введенні індикатора R, що вказує на те, чи значення конкретної змінної присутнє ($\text{R} = 1$), чи відсутнє ($\text{R} = 0$). Таким чином, кожне значення в наборі даних є парою спостережень – власне, саме значення (присутнє або відсутнє), та відповідне R. Визначення пропущених даних як окремої змінної створено для того, щоб вказати, що R може мати ймовірнісний розподіл. Розподіл залежить від причини пропусків, що зазвичай не є відомою, тому й встановити параметри розподілу не є можливим. Однак важливо те, що розподіл присутній і ймовірність може залежати або не залежати від інших змінних у даних. Це дозволяє точніше формалізувати механізми пропущених даних. Так, для MNAR типу розподіл можна записати як:

$$p(R|Y_{\text{с}}, Y_{\text{п}}, \phi)$$

де R – індикатор пропуску, $Y_{\text{с}}$ та $Y_{\text{п}}$ це спостережувані та пропущені дані, а $\phi$ – параметр або набір параметрів, що описують зв'язок між R та даними. Таким чином, ймовірність значення R може залежати як від інших змінних ($Y_{\text{с}}$), так і від значень самого $Y$ ($Y_{\text{п}}$).

Механізм MAR можна описати як

$$p(R|Y_{\text{с}}, \phi)$$

тобто ймовірність пропуску залежить від спостережуваної інформації і залежність описана деяким параметром $\phi$.

Нарешті, механізм MCAR описується виразом

$$p(R|\phi)$$

що вказує на те, що існує деякий параметр, що визначає значення R, але цей параметр більше не залежить від даних.

Механізми утворення пропущених даних є важливими, оскільки вони визначають як працюватиме той чи інший метод заповнення на конкретних

даних. Методи потребують виконання певного припущення для коректної роботи – тобто для того, щоб створювати дані, які не будуть мати зміщені статистичні параметри. Прості традиційні методи заповнення, такі як аналіз повних спостережень, заповнення середнім, медіаною, потребують MCAR припущення і даватимуть зміщені результати на інших механізмах. Потужніші методи заповнення даних, такі як метод максимальної правдоподібності або метод множинного заповнення, потребують виконання MAR припущення. Стосовно MAR припущення може виникнути питання чи таке припущення взагалі може бути правдоподібним, якщо, як зазначалося вище, точно перевірити, чи справджується MAR припущення, неможливо. В дійсності MAR припущення справджується не завжди, однак не всі його порушення є однаково шкідливими для методів заповнення. Визначення MNAR вказує на те, що існує взаємозв'язок між ймовірністю наявності пропусків у $Y$ та самого значення $Y$. Цей зв'язок може з'являтися з двох причин. Перша – дійсно можливо, що ймовірність наявності пропуску прямо пов'язана з значенням. Проте такий зв'язок також може з'являтись коли обидва ці параметри пов'язані з деякою змінною, що не вимірюється і не належить до спостережуваних даних. Якщо така змінна буде додана до дослідження, то MNAR механізм буде перетворено на MAR. Загалом це вказує на необхідність ретельного дослідження причин наявності пропусків.

Також для збільшення ймовірності досягнення виконання MAR припущення корисним рішенням може бути введення допоміжних змінних. Допоміжна змінна – це змінна, що корелює з пропусками або з неповними даними по змінній. Допоміжні змінні зазвичай не є цікавими для дослідника (тобто вони не були б додані якби дані були повні) і їх основна роль це полегшити процес аналізу пропущених даних та підвищити ймовірність досягнення виконання MAR припущення, що позитивно вплине на роботу методів заповнення. Зазвичай

обчислювальна ціна додавання допоміжних змінних є невисокою навіть коли їх є велика кількість, тому рекомендується додавати їх стільки, скільки потрібно.

Окремим частковим випадком є дизайн спланованих пропусків, що, зокрема, використовується в організації опитувань. Сплановані пропуски можуть автоматично створювати MCAR або MAR механізми, однак в цьому випадку у дослідника є повний контроль над пропусками і їх аналіз це окремий випадок. Дизайн спланованих пропусків використовує відсутні дані для вирішення ряду практичних завдань. Зокрема, сплановані пропуски можуть зменшувати складність опитувань для респондентів, знижувати вартість, пов'язану із збором даних, та зменшувати складність збору даних у довготривалих дослідженнях. Складні методи заповнення, такі як метод максимальної правдоподібності та метод множинного заповнення, дозволяють дослідникам аналізувати дані з спланованими пропусками без необхідності відкидати неповні випадки, а втрата потужності внаслідок відсутніх даних, як правило, не є пропорційною рівню відсутніх даних.

## 1.3 Попередня обробка даних

Попередня обробка даних є однією з ключових стадій підготовки даних перед навчанням моделей машинного навчання. Ця стадія грає важливу роль у підготовці даних для аналізу та моделювання, і включає в себе низку кроків та перевірок.

1. Перевірка на пропущені дані: першим кроком є ідентифікація пропущених даних у наборі даних. Цей крок є найпершим, оскільки всі наступні дії потребують повного набору даних. Без кроку перевірки пропусків та їх заповнення наступні дії втратять сенс. Крім того, немає сенсу виконувати

заповнення після інших дій, оскільки ці дії змінюють значення і їх прийдеться повторно виконувати для кожного заповненого значення.

Незалежно від наявності пропусків (та відповідної необхідності заповнення) після першого етапу варто провести аналіз даних. Аналіз даних полягає в глибокому вивченні та отриманні розуміння структури набору даних перед застосуванням будь-яких аналітичних або статистичних методів. Основна мета цього процесу – це виявлення важливих взаємозв'язків, закономірностей та особливостей у даних. Це допомагає визначити, які ознаки можуть бути корисними для моделювання та прогнозування, які можуть бути вилучені або змінені, а також виявити аномальні значення. Однією з ключових складових аналізу є візуалізація даних. Побудова графіків та діаграм дозволяє дослідити розподіл ознак, кореляційні зв'язки та інші важливі аспекти даних. Візуалізація сприяє кращому розумінню даних і допомагає з визначенням того, які дії необхідно проробити з якими змінними, які аналітичні методи можуть бути найбільш ефективними, які прогнозні моделі будуть ймовірно мати кращі результати. Також проводять статистичний аналіз, обчислюють основні статистичні показники, вивчають розподіли ознак, створюють та перевіряють гіпотези. Цей процес допомагає збільшити якість даних та підготувати їх для подальшого використання в моделях машинного навчання. Проведений аналіз дозволяє отримати краще розуміння взаємозв’язків між ознаками, що може допомогти з визначенням наявного механізму утворення пропусків.

2. Заповнення пропущених даних: після ідентифікації пропущених даних слід вирішити, як їх заповнювати. Цей етап, в залежності від кількості та складності пропусків, може бути як доволі простим, так і досить складним. Після проведення аналізу даних, обрання методу та виконання заповнення можна переходити на наступні кроки.

3. Масштабування даних: застосовується коли різні спостережувані змінні в наборі даних мають суттєво різний діапазон значень: наприклад, в однієї змінної значення коливаються в діапазоні від 0 до 1, а в іншої – від 0 до 1000. Масштабування даних до стандартних діапазонів, наприклад від 0 до 1 або за допомогою стандартизації (з дисперсією 1 та середнім значенням 0), допомагає моделям швидше збігатися та підвищує точність передбачень. Відсутність масштабування призведе до того, що, змінна з більшими значеннями гратиме більшу роль у навчанні моделі, що є небажаним, оскільки це ймовірно матиме негативний вплив на результати навчання та відповідну точність моделі.
4. Семплінг: технологія створення штучних даних, що використовується у разі нерівномірного розподілу класів або великої кількості даних. Завдання семплінгу для нерівномірного розподілу класів – збалансувати набір даних для поліпшення результатів навчання. Для випадку великої кількості даних - зменшити обчислювальні витрати.
5. Кодування категоріальних даних: категоріальні дані (наприклад, розділи або типи) потребують кодування у числовому форматі, щоб бути придатними для використання у моделюванні. Кодування може бути досягнуто за допомогою методів one-hot encoding або label encoding.
6. Видалення зайвих ознак: неінформативні, корельовані або дубльовані ознаки можуть бути видалені, щоб покращити продуктивність моделі та зменшити ризик перенавчання. Визначити такі ознаки можна шляхом проведення статистичного аналізу наявних даних, візуалізації даних, тощо.
7. Нормалізація даних: може включати в себе нормалізацію текстових даних, зменшення шуму та інші методи обробки, спрямовані на поліпшення якості та готовності даних для моделювання.

Завершення процесу обробки даних дозволяє перейти до наступних етапів розробки моделей прогнозування.

## 1.4 Класифікація методів заповнення

Дослідники вивчають проблему пропущених даних десятиліттями і за цей час було запропоновано десятки технік для вирішення цієї проблеми. Багато з цих підходів користуються широкою популярністю, тоді як інші використовуються більше з навчальною метою. Найбільш поширені методи можна умовно назвати "традиційними". Загалом вони широко поширені, прості до використання та виконують найважливіше завдання методів заповнення – створюють повний набір даних, що придатний до подальшого використання. Проте вони також мають ряд серйозних недоліків, через що їх використання часто може сильно спотворити дані та створити додаткові проблеми.

Окремо традиційні методи можна розділити на методи заповнення та видалення даних. Методи видалення видаляють дані з набору коли в них присутні пропуски - це підхід, що присутній в статистичних програмних пакетах і дуже поширений в деяких дисциплінах, таких як психологія і освіта. Ці методи працюють лише на MCAR механізмі, інакше вони будуть призводити до спотворення оцінок параметрів. Проте навіть за умови виконання MCAR припущення методи видалення даних завдають більше шкоди даним, ніж користі, через що їх використання є небажаним.

На відміну від простих методів заповнення, складні методи надають можливість ефективно вирішувати проблему відсутності даних завдяки здатності заповнювати пропущені дані без спотворення. Вони потребують MAR припущення для коректної роботи. Одним із найпоширеніших та потужних

методів є метод множинного заповнення, який дозволяє створити кілька варіантів набору даних з заповненими значеннями, після чого проаналізувати отримані набори та об’єднати їх. Іншим потужним методом, що використовується коли для дослідника важливіше отримати коректні статистичні дані, а не заповнити пропуски, є метод максимальної правдоподібності. Він базується на знаходженні найбільш ймовірних параметрів моделі, що найкраще підходять до спостережень.

Проте, важливо розуміти, що насправді, кращий спосіб вирішення проблеми пропущених даних з статистичної точки зору – це не мати пропущених даних взагалі, оскільки навіть найкращі з методів мають власні похибки. Тому в інтересах дослідників за можливості намагатися отримати настільки повні дані у своєму дослідженні, наскільки це можливо, адже повні дані завжди є більш надійними для створення висновків.

## 1.5 Висновки до розділу

В цьому розділі було розглянуто актуальність дослідження та визначено, що задача заповнення пропусків даних має високу актуальність в контексті попередньої обробки даних при роботі з прогнозними моделями машинного навчання. Механізми утворення пропусків у даних поділяються на три типи в залежності від наявності взаємозв’язків між даними. Існує велика кількість різноманітних методів заповнення, що потребують виконання певного припущення про механізм утворення пропусків для своєї коректної роботи. У наступному розділі роботи детально розглядаються поширені методи заповнення даних, їх переваги та недоліки, особливості використання.

# РОЗДІЛ 2 МОДЕЛІ ТА МЕТОДИ ВИКОНАННЯ ДОСЛІДЖЕННЯ

## 2.1 Традиційні методи заповнення пропущених даних

### 2.1.1 Методи видалення даних

Методи аналізу повних спостережень (англ. listwise deletion) та аналізу наявних спостережень (англ. pairwise deletion) є найбільш поширеними методами обробки втрачених даних в багатьох галузях наук. Головною перевагою цих методів є їх зручність використання. Однак ці методи видалення мають серйозні обмеження, які роблять їх використання невигідним в більшості випадків. Найважливішим є те, що ці підходи передбачають виконання MCAR припущення про пропущені дані та можуть спотворювати оцінки параметрів, коли це припущення не виконується. Проте навіть якщо припущення MCAR є правдоподібним, видалення даних є шкідливим і може значно зменшити потужність аналізу та кількість наявної інформації, що використовуватиметься в дослідженні. Через це використання цих методів загалом не є рекомендованим, особливо зважаючи на те, що простота використання – яка є основною перевагою цих методів – не є вирішальним фактором коли велика кількість програмного забезпечення містить реалізації складніших методів, що дозволяють отримати набагато кращі результати без істотного підвищення рівня складності використання.

Аналіз повних спостережень видаляє весь запис, якщо хоча б одна з ознак є пропущеною. Цей метод є зручним і швидким в реалізації та в результаті дає повний набір даних. У більшості випадків недоліки цього методу перекривають всі його переваги. Основна проблема полягає в тому, що для його застосування необхідні дані MCAR, і він може спотворювати оцінки параметрів, коли це

припущення не виконується. Як приклад, нехай дані по змінній $X$ відсутні коли значення по змінній $Y$ переважають певний поріг $a$ – у цьому випадку механізмом пропусків є MAR і, якщо застосувати аналіз повних спостережень, то оцінки середніх значень будуть зміщеними (занадто малими або високими). Таким чином, спостереження, що залишились після застосування аналізу повних спостережень неточно представляють гіпотетично повний набір даних, оскільки вони мають систематично інші значення змінних. Крім того, обмеження діапазону, яке виникає внаслідок викидання хвостів розподілів, обмежує варіативність даних і зменшує величину кореляції.

Навіть без урахування проблеми зміщення оцінок, аналіз повних спостережень може бути дуже марнотратним, особливо коли видалені спостереження містять дані щодо великої кількості змінних. Видалення неповних записів може призвести до суттєвого зменшення загального розміру вибірки, величина якого зростає зі збільшенням відсотку пропущених даних або кількості змінних. Наприклад, розглянемо набір даних із 10 змінних, де кожна має відсутність приблизно 2% спостережень в повністю випадковому порядку. Хоча відсоток втрат даних для кожної окремої змінної є відносно невеликим, аналіз повних спостережень в середньому призведе до видалення приблизно 18% записів. З 20 змінними очікуваний відсоток повних спостережень знижується приблизно до 67%. Не дивно, що це скорочення розміру вибірки може суттєво знизити статистичну потужність, особливо при малих та середніх вибірках. Зниження потужності для цього методу є проблемою завжди, навіть у найкращому випадку, коли дані є MCAR.

Метод аналізу наявних спостережень намагається зменшити втрату даних, видаляючи випадки на основі кожного окремого аналізу. Прикладом використанням попарного вилучення є випадок, коли дослідник використовує різні підмножини спостережень для обчислення кожного елемента матриці

кореляцій. Використання якнайбільшої кількості можливих даних безперечно є хорошою ідеєю, і зазвичай аналіз наявних спостережень є більш потужним, ніж аналіз повних спостережень, особливо коли змінні в наборі даних мають низькі або помірні кореляції. Однак недоліки цього методу обмежують його користь. Як і аналіз повних спостережень, цей метод вимагає виконання MCAR припущення для роботи. Іншою проблемою є те, що кожне обчислене статистичне значення може базуватися на різних підмножинах даних. Для прикладу розглянемо наступну формулу для вибіркової коваріації:

$$\hat{\sigma}_{XY} = \frac{\sum(x_i - \hat{\mu}_X)(y_i - \hat{\mu}_Y)}{N - 1}$$

Аналіз наявних спостережень використовує підмножину повних спостережень по обом змінним $X$ та $Y$ для обчислення коваріації. Цю ж підмножину можна використати для обчислення вибіркових середніх $\hat{\mu}_X$ та $\hat{\mu}_Y$, але також для їх обчислення можуть бути використані відповідні підмножини записів з повними значеннями $X$ та $Y$. Схожу проблему можна зустріти і при обчисленні знаменника для коефіцієнта кореляції:

$$r = \frac{\hat{\sigma}_{XY}}{\sqrt{\hat{\sigma}_X^2 \hat{\sigma}_Y^2}}$$

Оскільки зазвичай для обчислення дисперсій $\hat{\sigma}_X^2$ та $\hat{\sigma}_Y^2$ використовується підмножина повних даних по $X$ та $Y$, однак також можна використати відповідні підмножини даних з повними значеннями $X$ та $Y$.

### 2.1.2 Методи одиночного заповнення

Другою категорією традиційних методів заповнення є методи одиночного заповнення – термін походить від того, що ці методи створюють єдине значення для заповнення кожного пропуску, що відрізняє їх від складніших методів, таких як метод множинного заповнення, що створює декілька копій даних і на основі їх аналізу створює єдине заповнення. Заповнення - приваблива стратегія, оскільки воно дає повний набір даних в результаті. . На перший погляд, заповнення також вигідне тим, що воно використовує дані, які методи видалення в іншому випадку відкидали б. Незважаючи на ці очевидні переваги, більшість методів одиночного заповнення мають потенційно серйозні недоліки. Як і методи видалення більшість методів одиночного заповнення дають зміщені оцінки параметрів, навіть в ідеальній ситуації, коли дані відносяться до MCAR типу. Крім того, одноразові техніки заміщення зменшують стандартні помилки. На інтуїтивному рівні відсутні значення мають збільшувати стандартні помилки, оскільки вони додають ще один рівень шуму до оцінок параметрів. Однак, аналіз набору даних, що був заповнений методом одиночного заповнення, розглядає заповнені значення як реальні дані, тому навіть найкраща з цих методів буде недооцінювати помилку вибірки.

Найпершим методом, що варто розглянути, є заповнення середнім значенням. Це доволі інтуїтивний метод, що використовує середнє значення наявних даних по кожній ознаці для заповнення пропущених даних у ній. Проте, цей метод істотно спотворює дані навіть у випадку виконання MCAR припущення. Використання середнього значення в розподілі доволі очевидно зменшує варіативність даних, що, в свою чергу, зменшує стандартне відхилення та дисперсію, а також амплітуди коваріацій та кореляцій. Ці проблеми присутні

завжди, незалежно від наявного механізму пропущених даних і чим більше є пропусків, тим більшими є зміщення.

Іншим поширеним методом є метод регресійного заповнення. Він замінює відсутні значення передбаченими з рівняння регресії. Основна ідея за цим підходом інтуїтивно приваблива: використання інформації з повних змінних для заповнення неповних змінних. Змінні часто є корельованими, тому раціонально генерувати заповнення, що використовує інформацію зі спостережуваних даних. Фактично, взяття інформації зі спостережуваних даних - це стратегія, яка використовується у складних методах, таких як метод максимальної правдоподібності та множинного заповнення, хоча останні роблять це більш складним способом. Перший етап процесу заповнення полягає в оцінці набору рівнянь регресії, які генеруватимуть пропущені значення на основі наявних. Зазвичай ці оцінки генеруються за допомогою аналізу на основі повних випадків. Другий етап полягає у створенні передбачених значень для змінних з пропусками. Ці передбачені значення заповнюють відсутні значення та створюють повний набір даних.

Окремою складністю заповнення регресією є те, що під кожну комбінацію пропусків необхідно створювати окреме регресійне рівняння. Наприклад, якщо взяти набір даних з трьома спостережуваними змінними $Y_1, Y_2, Y_3$, в кожній з яких є пропущені дані. Таким чином, є 6 комбінацій пропусків:

1. Лише $Y_1$
2. Лише $Y_2$
3. Лише $Y_3$
4. $Y_1$ та $Y_2$
5. $Y_1$ та $Y_3$
6. $Y_2$ та $Y_3$

З зростанням кількості спостережуваних змінних також швидко зростатиме і кількість комбінацій пропусків і, відповідно, кількість рівнянь. Простий спосіб побудувати рівняння - почати з оцінки вектору середніх і матриці коваріації, оскільки елементи в цих матрицях визначають всі необхідні коефіцієнти регресії. Вибіркові параметри, як і раніше, обчислюються з підмножини повних спостережень. В результаті підстановка наявних значень у відповідні рівняння регресії створює значення для пропусків, якими здійснюється заповнення, що формує повний набір даних.

Регресійне заповнення краще за заповнення середнім, але має схожі передбачувані проблеми. У випадку двох змінних заповнені значення потрапляють чітко на регресійну пряму із ненульовим нахилом, що вказує на те, що кореляція між змінними для певної підмножини спостережень дорівнює 1. У випадку багатовимірної регресії те саме можна спостерігати з площиною. Таким чином, регресійне заповнення фактично стикається з прямо протилежною проблемою у порівнянні з заповненням середнім, оскільки воно заповнює дані ідеально корельованими показниками. У багатовимірних наборах даних заповнені значення не матимуть ідеальних кореляцій з іншими змінними, але кореляції все одно будуть високими. Отже, заповнення регресією завищує кореляції навіть коли дані є MCAR. Той факт, що заміщені значення лежать безпосередньо на прямій лінії (або площині у випадку багатовимірності даних), вказує на те, що заповнені дані не мають варіативності, що була б присутня, якби дані були повними.

Стохастичне регресійне заповнення – це метод, що також використовує рівняння регресії, але має додатковий крок у вигляді доповнення кожного згенерованого значення нормально розподіленим залишком. Додавання залишків до заповнених значень відновлює варіативність даних, завдяки чому стохастична регресія не має тих проблем, які має звичайне регресійне заповнення. Крім того,

стохастична регресія це один з небагатьох методів одиночного заповнення, що здатний створювати незміщені заповнення при MAR механізмі, завдяки чому він є одним з найбільш надійних з цієї категорії методів. Алгоритм роботи методи загалом ідентичний до алгоритму звичайної регресії, а залишок обчислюється як випадкове значення з нормального розподілу з середнім значенням 0 та дисперсією, що рівна залишковій дисперсії з відповідного регресійного рівняння. На перший погляд стохастичне регресійне заповнення виглядає як робоча альтернатива до складних алгоритмів, про які йтиметься далі, але, як і інші методи одиночного заповнення, стохастична регресія зменшує стандартні помилки, що збільшує ймовірність помилок першого роду. Таким чином, стохастична регресія, хоч і є доволі якісним методом, у порівнянні з іншими методами одиночного заповнення, також має власні проблеми, що зменшують її користь у порівнянні з більш складними методами.

Hot-deck заповнення – це група методів заповнення, основна ідея яких полягає у використанні в якості заповнення значень, що походять з схожих спостережень. Найбільш типовий варіант цього методу полягає у тому, щоб для кожного пропуску взяти випадкове значення з підмножини повних спостережень, що мають схожі значення за певним набором змінних. Ці методи, зазвичай, не призводять до погіршення варіативності даних до такого ж рівня, як інші методи одиночного заповнення, але вони можуть створювати зміщення у оцінках кореляцій та коефіцієнтів регресії.

Окремо можна розглянути метод заповнення, що використовується у довготривалих дослідженнях – заповнення на основі останнього спостереження (англ. Last Observation Carried Forward). Процедура заповнення у цьому методі полягає у використанні останнього значення перед появою пропусків, яким заповнюються усі наступні пропуски, що йдуть підряд. Метод може застосовуватися як до випадків повної відсутності значень починаючи з якогось

моменту, так і до випадків часткових пропусків. Загалом вважається, що заповнення на основі останнього спостереження дає консервативну оцінку показників після завершення дослідження, оскільки воно вводить в дані значення, що не змінюються з часом. Проте дослідження показують, що це не завжди так. Передбачити масштаб та напрям зміщень складно, оскільки це сильно залежать від конкретних характеристик даних, але використання цього методу швидше за все призведе до спотворених оцінок параметрів, навіть коли дані є MCAR.

## 2.2 Складні методи заповнення пропущених даних

Складні методи заповнення пропущених даних, такі як множинне заповнення, метод максимальної правдоподібності та інші, мають численні переваги над традиційними методами заповнення. В першу чергу, вони орієнтуються на MAR механізм даних, що є недосяжним для більшості традиційних методів. Вони дозволяють використовувати більше інформації з наявних даних, що покращує точність оцінок та підвищує статистичну потужність аналізу, їх ідеї враховують та вирішують потенційні проблеми з заповненням – зміщення оцінок статистичних параметрів, зменшення стандартних помилок, зменшення варіативності, зменшення потужності. Ці методи складніші у теоретичному плані, а також у реалізації та використанні, але результат їх роботи часто суттєво кращий, ніж у традиційних методів. Крім того, багато сучасних програмних пакетів для аналізу даних містять у собі реалізації цих методів, використання яких не є суттєво складнішим, ніж використання традиційних методів.

### 2.2.1 Метод максимальної правдоподібності

У першу чергу варто розглянути метод максимальної правдоподібності. У загальному випадку це метод оцінювання значення змінної шляхом максимізації функції правдоподібності. Метою методу максимальної правдоподібності є визначення параметрів повного набору даних, які мають найвищу ймовірність створення наявних вибіркових даних. Вибіркова логарифмічна функція правдоподібності відіграє центральну роль у цьому процесі, оскільки вона кількісно визначає відносну ймовірність вибору множини значень з нормального розподілу з певним вектором середніх та матрицею коваріації.

Підстановка значення показника (або набору показників) в функцію густини ймовірності повертає значення логарифмічної функції правдоподібності для окремого випадку, і вибіркова логарифмічна функція правдоподібності є сумою окремих її значень. Вибіркова логарифмічна функція правдоподібності відіграє роль показника відповідності між даними та оцінками параметрів і надає основу для вибору серед набору правдоподібних значень параметрів.

Концептуально, оцінка - це ітеративний процес, що повторно перевіряє різні значення параметрів до тих пір, поки не знайде оцінки, які найбільш імовірно могли б створити такі дані. Ітеративні алгоритми попередньо визначають вихідні параметри та використовують їх для обчислення функцій втрат чи градієнтів, вдосконалюючи параметри на кожному кроці. У випадку методу максимальної правдоподібності в якості функції використовується логарифмічна функція правдоподібності

$$L(\theta \mid x) = lnf_x(\theta \mid x)$$

Процедура оцінки по суті повторює обчислення логарифмічної функції правдоподібності багато разів, кожного разу підставляючи різні значення параметрів сукупності в рівняння функції. Оскільки логарифми є строго

зростаючими функціями, максимізація ймовірності еквівалентна максимізації логарифма ймовірності. Проте з практичних міркувань зручніше працювати з логарифмічною функцією ймовірності, оскільки її обчислення зазвичай простіше та вона дозволяє легше порівнювати різні оцінки завдяки тому, що звичайна функція правдоподібності повертає малі значення отримані з добутків, коли логарифмічна повертає великі суми.

Кожна унікальна комбінація оцінок параметрів дає різне значення функції правдоподібності, і метою оцінки є визначення конкретної комбінації оцінок, яка дає найвище значення правдоподібності і, отже, найкраще відповідає наявним даним. Найперші ітерації продукують найбільші зміни у значенні логарифмічної функції правдоподібності, останні ітерації практично не змінюють її значення. Коли стає помітно, що подальші ітерації не збільшують значення функції робота алгоритму припиняється, оскільки він досяг значень, з якими отримується найбільше значення логарифмічної функції правдоподібності.

У деяких ситуаціях похідні логарифмічної функції правдоподібності породжують відомі рівняння, які визначають оцінки методу максимальної правдоподібності, але більш складні ситуації застосування методу максимальної правдоподібності (включаючи роботу з пропущеними даними) вимагають ітеративних алгоритмів оптимізації для визначення найбільш імовірних значень параметрів.

Кривизна логарифмічної функції правдоподібності надає важливу інформацію щодо неоднозначності оцінки. Плоска функція правдоподібності ускладнює вибір між конкуруючими оцінками, оскільки значення журналу правдоподібності відносно схожі для різних оцінок параметрів. Натомість, крута функція правдоподібності більш чітко відокремлює найбільш правдоподібну оцінку від інших можливих оцінок параметрів. Математично друга похідна визначає кривизну логарифмічної функції правдоподібності. Другі похідні в

основному визначають стандартні помилки методу найбільшої правдоподібності, таким чином, більші другі похідні (тобто графіки з більшою вершиною) вказують на менші стандартні помилки, а менші другі похідні (тобто більш плоскі функції) вказують на більші стандартні помилки.

Варто відмітити, що метод максимальної правдоподібності не вирішує задачу пропущених даних у стандартному сенсі, оскільки він не створює безпосередні заповнення. Ідея методу полягає у визначенні оцінок статистичних параметрів для змінних повного набору даних не виконуючи заповнення. Таким чином, повний набір даних не створюється в процесі роботи алгоритму. Цей факт грає важливу роль при виборі методу заповнення під конкретну задачу, оскільки в деяких задачах наявність повного набору даних не є настільки важливою як наявність точних оцінок цих даних, а в інших задачах оцінки грають меншу роль, ніж наявність повних даних. Зокрема, у контексті розв’язання проблеми пропусків при підготовці даних до навчання прогнозної моделі наявність повного набору даних є необхідністю, тому метод максимальної правдоподібності не є робочим варіантом.

Пропущені дані вводять деякі додаткові нюанси, які не є важливими при використанні методу для повних даних. По-перше, неповні записи даних вимагають невеликої зміни в обчисленнях логарифмічної функції правдоподібності на рівні кожного окремого спостереження, щоб врахувати той факт, що записи вже не мають однакову кількість спостережених точок даних. Втрачені дані також потребують тонкої, але важливої, корекції в обчисленнях стандартної помилки. Зазвичай аналіз втрачених даних вимагає ітеративних алгоритмів оптимізації, навіть для дуже простих задач оцінки.

Обчислення стандартної похибки починаються з матриці других похідних, тобто матриці Гессе. Множення матриці Гессе на -1 створює матрицю інформації, а обчислення оберненої матриці до матриці інформації дозволяє отримати

матрицю коваріації параметрів. Діагональні елементи матриці коваріації параметрів містять вибіркові дисперсії оцінок параметрів, і взяття кореня з цих елементів дозволяє отримати стандартні помилки.

Один з найбільш поширених ітеративних алгоритмів для методу максимальної правдоподібності це EM-алгоритм. EM-алгоритм - це двоетапний ітеративний процес, який складається з кроку E-Expectation та кроку M-Maximization. Ітеративний процес розпочинається з початкової оцінки вектору середніх та матриці коваріації (наприклад, за допомогою оцінки з використанням аналізу повних спостережень). Крок E використовує елементи вектору середніх та матриці коваріації для створення набору рівнянь регресії, які передбачають неповні змінні на основі спостережуваних змінних. Мета кроку E - заповнити пропущені значення схожим чином до заповнення стохастичною регресією. Як і у заповненні регресією, EM потребує окремі регресійні рівняння для кожної комбінації пропусків при роботі з багатовимірними наборами даних. Після виконання заповнення крок M застосовує стандартні формули для повних даних до заповнених даних для генерації оновлених оцінок вектору середніх та матриці коваріації. Алгоритм переносить оновлені оцінки параметрів на наступний крок E, де він створює новий набір рівнянь регресії для передбачення відсутніх значень. Наступний крок M повторно оцінює вектор середніх та матрицю коваріації. EM повторює ці два кроки, доки оцінки більше не змінюються між послідовними кроками M, після чого алгоритм збігається на оцінці максимальної правдоподібності.

### 2.2.2 Метод множинного заповнення

Аналіз за допомогою методу множинного заповнення складається з трьох окремих етапів: фази заповнення, фази аналізу та фази об'єднання. На етапі заповнення використовується алгоритм доповнення даних — це двоетапна процедура, яка складається з кроку заповнення I та апостеріорного кроку P. I-крок ідентичний алгоритму стохастичної регресії, зокрема, I-крок використовує оцінку вектору середніх та коваріаційної матриці для побудови набору рівнянь регресії, які передбачають значення пропущених даних з спостережуваних змінних. Для кожної конфігурації змінних з пропущеними значеннями створюється окреме регресійне рівняння. Прогнозовані значення потрапляють безпосередньо на лінію регресії (або поверхню регресії, у багатовимірному випадку), тому додавання нормально розподіленого залишкового члена до кожного прогнозованого значення відновлює варіативність заповнених даних. З точки зору Байєсівського підходу, заповнені значення є випадковими значеннями з умовного розподілу, що залежить від спостережуваних даних і оцінок вектору середніх та коваріаційної матриці на конкретному I-кроці. Формально I-крок можна описати наступним рівнянням:

$$Y_t^* \sim p(Y_{\text{пр}}/Y_{\text{сп}}, \theta_{t-1}^*)$$

де

$Y_t^*$ - заповнені значення на I-кроці

$Y_{\text{пр}}$- пропущені дані

$Y_{\text{сп}}$- спостережувані дані

$\theta_{t-1}^*$- позначає вектор середніх та коваріаційну матрицю з попереднього P-кроку.

Іншими словами, рівняння вказує на те, що заповнені значення на конкретному I-кроці є випадково взятими з розподілу правдоподібних замінюваних значень, що залежить від спостережуваних даних і поточних оцінок параметрів.

Кінцевою метою етапу заповнення є створення m повних наборів даних, кожен з яких містить унікальні оцінки відсутніх значень. Створення унікальних заповнень потребує різних оцінок коефіцієнтів регресії на кожному I-кроці, а метою P-кроку є створення альтернативних оцінок вектору середніх та коваріаційної матриці (необхідні компоненти для рівнянь регресії на I-кроках).
P-крок починається з використання заповнених даних з попереднього I-кроку для оцінки вектору середніх значень і коваріаційної матриці. Формально P-крок описується так:

$$\theta_t^* \sim p(\theta / Y_{\text{сп}}, Y_t^*)$$

де

$\theta_t^*$ - симульовані значення параметрів на P-кроці (вектор середніх та коваріаційна матриця)

$Y_{\text{сп}}$- спостережувані дані

$Y_t^*$ - включає заповнені значення з попереднього I-кроку

Тобто, симульовані значення параметрів P-кроку є випадково взятими значеннями з розподілу, що залежить від спостережуваних даних та даних, що були заповнені на попередньому I-кроці. Далі алгоритм генерує новий набір значень параметрів шляхом додавання випадкового залишкового члена до кожного елемента в векторі середніх та коваріаційної матриці. Це схоже на отримання нового набору вірогідних оцінок із розподілу вибірки. Багаторазове повторення цієї двоетапної процедури створює набір копій даних, кожна з яких містить унікальні оцінки відсутніх значень.

Таким чином, на етапі заповнення створюється кілька копій набору даних (наприклад, m = 20), кожна з яких містить різні оцінки відсутніх значень.

Концептуально цей крок є ітераційною версією заповнення за допомогою алгоритму стохастичної регресії, хоча його математичні основи значною мірою покладаються на принципи оцінки Байєса. Як випливає з назви, метою фази аналізу є аналіз заповнених наборів даних. На цьому кроці застосовуються ті самі статистичні процедури, які були б використані, якби дані були повними. Єдина відмінність полягає в тому, що аналіз виконується окремо для кожного заповненого набору даних, тобто m разів. Таким чином, фаза аналізу дає m наборів оцінок параметрів і стандартних помилок, тому мета фази об'єднання полягає в тому, щоб об'єднати все в єдиний набір результатів.

Метод множинного заповнення явно враховує невизначеність, пов'язану з відсутніми даними. Завдяки багаторазовому заповненню даних множинне заповнення дає оцінки змінних, які є середніми для ряду правдоподібних значень заміни, тому процес ніколи не довіряє єдиному набору заповнень. Це є істотною відмінністю від попередньо розглянутих методів заповнення, які розглядають один набір заповнених значень як реальні дані.

Загалом метод множинного заповнення вважається одним з найбільш досконалих варіантів для розв'язання проблеми пропущених даних, оскільки він здатний генерувати коректні незміщені результати заповнення як для MCAR, так і для MAR механізмів пропусків, а також створювати повний набір даних, що є важливим фактором при виборі методу заповнення для роботи з наборами даних для навчання прогнозних моделей. Аналіз відсутніх даних є складним завданням, оскільки не існує правильної методологічної процедури. У багатьох, якщо не в більшості, випадках сліпе застосування методу множинного заповнення призводить до більш точного набору заповнень, ніж використання одного з традиційних методів обробки відсутніх даних. У цьому сенсі простий аналіз з припущенням MAR, ймовірно, кращий, ніж найкращий з традиційних підходів до роботи з відсутніми даними. Метод множинного заповнення все ще продукує

зміщені результати у випадку MNAR даних, але часто добре виконаний аналіз з припущенням MAR даних може бути кращим, ніж аналіз з припущенням MNAR, навіть якщо є підстави вважати, що відсутність запису систематично пов'язана з самою змінною. Це пов'язано з тим, що моделі MNAR найчастіше покладаються на ще менш надійні припущення, ніж моделі MAR.

Визначення того, які змінні включити до використання у заповненні, є важливим аспектом множинного заповнення. Як мінімум, процес заповнення повинен включати всі змінні, що планується використовувати у подальшому статистичному аналізі. Виключення таких змінних з заповнення призведе до послаблення їх взаємозв'язків з іншими змінними, навіть якщо дані є MCAR або MAR. Це підкреслює важливість вибору змінних, оскільки неправильна модель заповнення  може бути причиною утворення зміщень, що не виникають при використанні методу максимальної правдоподібності. На щастя, включення занадто багатьох змінних у процес заповнення малоймовірно призведе до зміщень, тому зазвичай рекомендується включати стільки змінних, скільки потрібно. Основний недолік включення занадто великої кількості змінних - це можливі проблеми зі збіжністю. Максимальна кількість змінних не може перевищувати кількість спостережень, але зазвичай кількість змінних є значно меншою, за кількість спостережень, тому це не є проблемою.

Алгоритм доповнення даних, що використовується на кроці заповнення, відноситься до методів Монте-Карло марковських ланцюгів. Метою цих методів є симуляція випадкової вибірки з розподілу. За допомогою створення ланцюга Маркова, у якому бажаний розподіл виступає як його рівноважний розподіл, можна отримати вибірку з бажаного розподілу, фіксуючи стани з цього ланцюга. У процесі роботи алгоритму доповнення даних I та P кроки створюють ланцюг з заповнених значень $Y_t^*$ та оцінок $\theta_t^*$. Після певної кількості кроків згенеровані заповнення отримуються з великого масиву правдоподібних оцінок параметрів

розподілу, тому значення $Y_t^*$ по суті вибираються з розподілу, який в середньому по всьому діапазону апостеріорного розподілу. Аналогічно для оцінок параметрів отриманий ланцюг значень формує апостеріорний розподіл, що береться в середньому по всіх можливих значеннях пропущених даних. Через це для алгоритму доповнення даних використовується інше визначення збіжності – якщо для алгоритму максимальної правдоподібності збіжність визначалася як відсутність зміни у оцінці значення з певної ітерації, то у випадку алгоритму доповнення даних збіжність визначається як досяжність стаціонарності розподілу. Дослідники часто оцінюють збіжність, визначаючи кількість ітерацій доповнення даних, які повинні пройти, перш ніж доповнення на ітерації $t + k$ стануть незалежними від тих на ітерації $t$. Спостереження за поведінкою симульованих значень параметрів протягом великої кількості кроків P - один з способів цього. Наприклад, якщо два набори оцінок параметрів $\theta_t^*$ та $\theta_{t+10}^*$ розділені 10 ітераціями доповнення і між ними є кореляція, то вона вказує на те, що апостеріорний розподіл систематично змінюється після 10 ітерацій. Отже, аналіз даних, які відокремлені всього лише 10 ітераціями доповнення, є неправильним, оскільки заповнені значення також залежать одне від одного. Якщо ж, нехай, для різниці у 50 ітерацій вказана кореляція відсутня, то можна зробити висновок, що $\theta_t^*$ та $\theta_{t+50}^*$ випливають із стійкого апостеріорного розподілу, тому ці два набори оцінок параметрів повинні давати незалежні доповнення. З практичної точки зору це означає, що принаймні 50 циклів доповнення даних повинні відокремлювати набори даних, які опрацьовуватимуться на наступному етапі аналізу.

Використання EM-алгоритму для оцінки вектору середніх та матриці коваріації є корисним попереднім етапом для аналізу множинного доповнення. Оцінки EM є хорошими початковими значеннями для алгоритму доповнення даних, оскільки вони зазвичай є представниками апостеріорного розподілу.

Внаслідок цього доповнення даних, як правило, буде збігатися швидше коли початкові значення обрані з набору початкових значень EM.

## 2.3 Висновки до розділу

В цьому розділі було розглянуто основні методи обробки пропущених даних, їх специфіку, особливості використання та ефективність. Розкрито переваги та недоліки поширених методів обробки пропущених даних.

Загалом методи поділені на дві категорії – традиційні та складні – і окремо традиційні поділені на методи видалення та методи одноразового заповнення. Зазвичай, традиційні методи є простими за своєю теоретичною складовою та особливостями використання, однак не здатні працювати з механізмами пропусків складнішими за MCAR та результати їх роботи часто мають статистичні спотворення. Складні методи зазвичай є більш комплексними, прямо розв’язують ряд поширених проблем з заповненням, завдяки чому здатні працювати з механізмом пропусків MAR та продукувати заповнення без спотворення даних.

Найбільш рекомендованим методом заповнення є множинне заповнення, оскільки цей метод є найбільш надійним про розв’язанні проблем пропущених даних за механізмами MCAR, MAR та іноді навіть MNAR. Перевагою методу є комплексність, що забезпечує детальний та якісний процес аналізу та заповнення даних і в результаті продукує повний набір даних.

# РОЗДІЛ 3 ОГЛЯД ПРОГРАМНОГО ПРОДУКТУ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

## 3.1 Обґрунтування вибору платформи та мови програмування

Прогнозування за допомогою методів машинного навчання у сучасному світі є надзвичайно важливим. Машинне навчання дозволяє аналізувати великі обсяги даних та виявляти приховані зв'язки і закономірності, що важко виявити за допомогою традиційних методів. Це відкриває безліч можливостей в різних сферах життя, від бізнесу та фінансів до науки, медицини та транспорту. Існує широке різноманіття мов програмування, що можуть використовуватися для роботи з алгоритмами машинного навчання: серед них R, Java, C++, Scala. Однак найбільш популярним вибором для розробки систем підтримки прийняття рішень на основі алгоритмів машинного навчання є мова програмування Python. Це потужна мова програмування, що підтримує об'єктно-орієнтоване програмування, функціональне програмування, має величезну кількість різноманітних пакетів та фреймворків для вирішення математичних та статистичних задач, для аналізу даних, а також для роботи конкретно з технологіями машинного навчання – наприклад, бібліотеки TensorFlow, PyTorch, scikit-learn та Keras пропонують багатий функціонал для навчання нейронних мереж, роботи з даними та візуалізації результатів.

Простота Python порівняно з іншими мовами програмування - це одна з його найбільших переваг. Завдяки читабельному і інтуїтивному синтаксису, Python стає ідеальним вибором для початківців і досвідчених програмістів. Відсутність складних і громіздких конструкцій дозволяє швидко створювати і розуміти код, а це зменшує час на розробку та підтримку програм, що в свою чергу дозволяє розробникам більше концентруватись на результатах та їх

покращенні. Python також має велику спільноту користувачів завдяки чому ця мова програмування та її супутні інструменти активно підтримуються та вдосконалюються. Фреймворки Python також надають можливість легко інтегрувати машинне навчання в інші програмні продукти та веб-додатки завдяки різним інтерфейсам та API. Ця гнучкість дозволяє розробникам створювати застосунки з використанням різних мов програмування, що робить Python гарним вибором для виконання широкого спектру завдань, в тому числі і у великих проектах.

Фреймворки машинного навчання Python підтримують широкий спектр методів попередньої обробки даних, щоб забезпечити якісну обробку і аналіз даних у задачах машинного навчання. В залежності від потреб користувач може обирати ті методи, що найбільше підходять до розв’язання його задачі.

Візуалізація в Python є невід'ємною частиною роботи з алгоритмами машинного навчання та аналізу даних. Python надає багато потужних бібліотек, таких як Matplotlib, Seaborn, Plotly та багато інших, які спрощують процес візуалізації даних. Вони дозволяють створювати різноманітні графіки, від лінійних графіків та гістограм до теплових карт і 3D-візуалізацій. Ці бібліотеки також надають можливість відслідковувати процес навчання моделей, відображати важливі метрики, які допомагають визначити ефективність моделі, і аналізувати результати прогнозувань. Візуалізація допомагає виявляти аномалії, розуміти вплив різних факторів на результати та приймати обґрунтовані рішення на основі отриманих даних. За допомогою інтерактивних графіків, можна взаємодіяти з даними та вносити корективи в аналіз на льоту.

Додатково, візуалізація в Python може бути використана для створення інтерактивних дашбордів та звітів, що дозволяє користувачам взаємодіяти з даними та отримувати аналітику у реальному часі. Ця можливість особливо корисна в бізнес-аналітиці та прийнятті рішень на основі даних. В цілому, в

Python є безліч інструментів та ресурсів для створення вражаючих візуалізацій, що полегшують роботу з даними та розуміння результатів машинного навчання. У роботі для візуалізації результатів було використано Python бібліотеки Matplotlib та Seaborn.

Враховуючи вищеперераховане для проведення дослідження впливу методів заповнення на результати роботи моделей прогнозування машинного навчання в якості основної мови програмування була обрана мова Python. Також у роботі обмежено використовується функціонал мови програмування R. Ця мова виникла у сфері статистики та аналізу даних, вона спеціалізується на роботі з даними та статистичному моделюванні. R також має багате різноманіття пакетів і бібліотек, що дозволяють виконувати різноманітні завдання в аналізі даних, від візуалізації і обробки даних до машинного навчання та роботи з геоінформацією. Вона популярна серед статистиків, науковців, аналітиків даних та дослідників, і використовується в багатьох галузях, включаючи епідеміологію, фінанси, біологію та багато інших.

Завдяки можливостям статистичного аналізу мова програмування R є доволі логічним вибором дослідників, що працюють з проблемами пропущених даних, оскільки її фреймворки пропонують широкий вибір методів заповнення. У даній роботі функціонал R використовується для підключення реалізації методу множинного заповнення з бібліотеки MICE (англ. Multiple Imputation by Chained Equation). Для полегшення процесу розробки також використовувався Anaconda - дистрибутив, який містить у собі Python, R та інші мови програмування, а також численні бібліотеки та середовища для роботи з даними, включаючи Jupyter Notebook. Це забезпечує легкість встановлення і керування різними версіями мов та пакетів, що робить Anaconda ідеальним для створення ізольованих середовищ для проектів машинного навчання, де можуть вимагатися різні версії бібліотек. Anaconda також надає зручний інтерфейс для створення

віртуальних середовищ, які дозволяють ізолювати проекти один від одного, забезпечуючи чистоту та стабільність. Однією з ключових переваг є також можливість використовувати Jupyter Notebook для інтерактивного аналізу та візуалізації даних. Це дозволяє швидко прототипувати код, ділитися результатами та зберігати документацію для проектів машинного навчання. Загалом Anaconda дозволяє суттєво прискорити процеси створення середовища розробки, що дозволяє розробнику приділити більше уваги безпосередньо розробці та роботі з результатами.

## 3.2 Постановка задачі та опис обраних компонентів

Для дослідження впливу різних методів заповнення даних на точність роботи моделі прогнозування в першу чергу необхідно обрати задачу прогнозування, що дозволить дослідити залежності. Тому загалом було обрано наступний алгоритм дій:

1. Вибір задачі прогнозування та набору даних для неї,
2. Загальний аналіз набору даних, висновки,
3. Штучне створення пропущених даних різного роду,
4. Вибір методів заповнення,
5. Виконання заповнення пропусків обраними методами, створення заповнених наборів даних,
6. Вибір алгоритмів прогнозування,
7. Попередня обробка кожного отриманого набору даних,
8. Навчання обраних алгоритмів на повному наборі даних,
9. Навчання обраних алгоритмів на кожному заповненому наборі даних,
10. Перевірка якості отриманих моделей на тестових вибірках,

11. Порівняння результатів роботи моделей, що навчалися на заповнених даних з результатами роботи моделей, що навчалися на початкових повних даних.

У якості задачі прогнозування була обрана задача визначення утримання та відтоку клієнтів банку та відповідний набір даних клієнтів. Аналіз рівня збереження клієнтів (англ. customer retention rate) є важливим аспектом стратегії будь-якого бізнесу. Ця метрика вимірює, наскільки ефективно компанія утримує своїх клієнтів та запобігає їх втраті. Зазвичай збереження існуючих клієнтів є набагато більш вигіднім, простим та дешевим, ніж приваблення нових. Через це дуже важливо аналізувати цей показник та досліджувати способи його поліпшення. Аналіз CRR допомагає ідентифікувати проблемні області та причини втрати клієнтів, що в свою чергу дозволяє вдосконалити стратегії утримання та підвищити якість обслуговування клієнтів. Високий рівень збереження клієнтів вказує на те, що компанія успішно будує відносини зі своїми клієнтами, забезпечуючи їхню лояльність.

Оскільки основною метою дослідження є робота з методами заповнення у поєднанні з прогнозними моделями стартовий набір даних обрано повним. Це дозволяє отримати результати прогнозування використовуючи повний набір даних, після чого штучним чином ввести пропуски різного роду, опрацювати їх та провести навчання моделей на заповнених наборах. Також повнота стартового набору дозволяє точно порівнювати статистичні параметри по кожній змінній між повним та заповненими наборами даних.

Обраний набір даних складається з 14 змінних:

- RowNumber: нумерація строки в наборі
- CustomerId: ідентифікатор клієнта в банківській системі
- Surname: прізвище клієнта
- CreditScore: кредитна оцінка клієнта

- Geography: країна походження клієнта
- Gender: стать клієнта
- Age: вік клієнта
- Tenure: кількість років які клієнт провів з банком
- Balance: баланс клієнта
- NumOfProducts: кількість продуктів банку, що використовує клієнт
- HasCrCard: індикатор наявності кредитної картки банку в клієнта
- isActiveMember: індикатор активності клієнта
- EstimatedSalary: оцінка заробітної платні клієнта
- Exited: цільова змінна, що відображає статус відтоку/збереження клієнта

Змінні RowNumber, CustomerId та Surname є ідентифікаційними і не несуть корисної інформації про діяльність клієнта, тому можуть бути видалені. Серед 11 змінних, що лишились, 4 є неперервними (Age, CreditScore, Balance, EstimatedSalary), 6 є категоріальними (Geography, Gender, Tenure, NumOfProducts, HasCrCard, IsActiveMember) та 1 є цільовою (Exited). Найбільш цікавими з точки зору дослідження якості заповнення є неперервні змінні, тому у подальшому в процесі штучного створення пропусків саме ці змінні будуть використовуватися найчастіше.

На етапі аналізу набору даних здійснюється досліджуються статистичні параметри змінних, розподіли, можливі кореляції між змінними. На рис. 3.1 наведені гістограми розподілів для цільової та неперервних змінних.


Рис. 3.1. Гістограми розподілів цільової та неперервних змінних

Дослідження показало, що набір даних є незбалансованим, оскільки кількість клієнтів, що залишились, становить близько 80%, відмовились від послуг лише 20%. Ця незбалансованість здатна ускладнити процес навчання моделі, тому на етапі попередньої обробки її необхідно буде виправити. Окрім цього, за графіками видно, що змінна Balance має нормальний розподіл, якщо не враховувати нульові значення балансу, а змінна EstimatedSalary має доволі рівномірний розподіл і швидше за все не здатна дати велику кількість корисної інформації.

На рис. 3.2 показані кореляції неперервних змінних.


Рис. 3.2. Кореляції неперервних змінних

Помітно, що важливих кореляцій між змінними немає.

На рис. 3.3 наведено графіки для неперервних змінних за цільовою змінною.

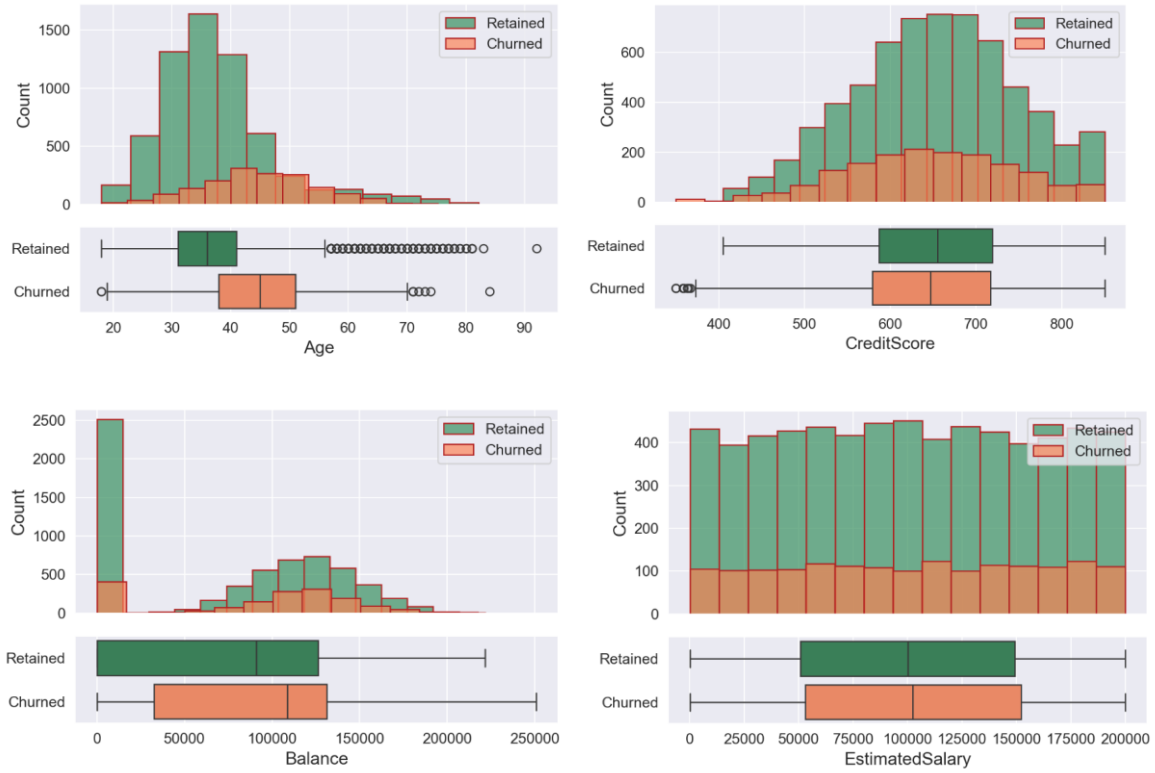


Рис. 3.3. Графіки неперервних змінних за цільовою змінною

З цих графіків стає видно, що існує помітна різниця між віковими групами, оскільки старші клієнти частіше відмовляються від послуг. Це може бути індикатором того, що стратегія роботи з старшою віковою групою не була адаптована під її потреби. За іншими змінними помічені приблизно однакові розподіли, що вказують на те, що ці змінні не мають істотного впливу на ймовірність відтоку клієнта.

Далі досліджуються категоріальні змінні, їх графіки відображені на рис. 3.4

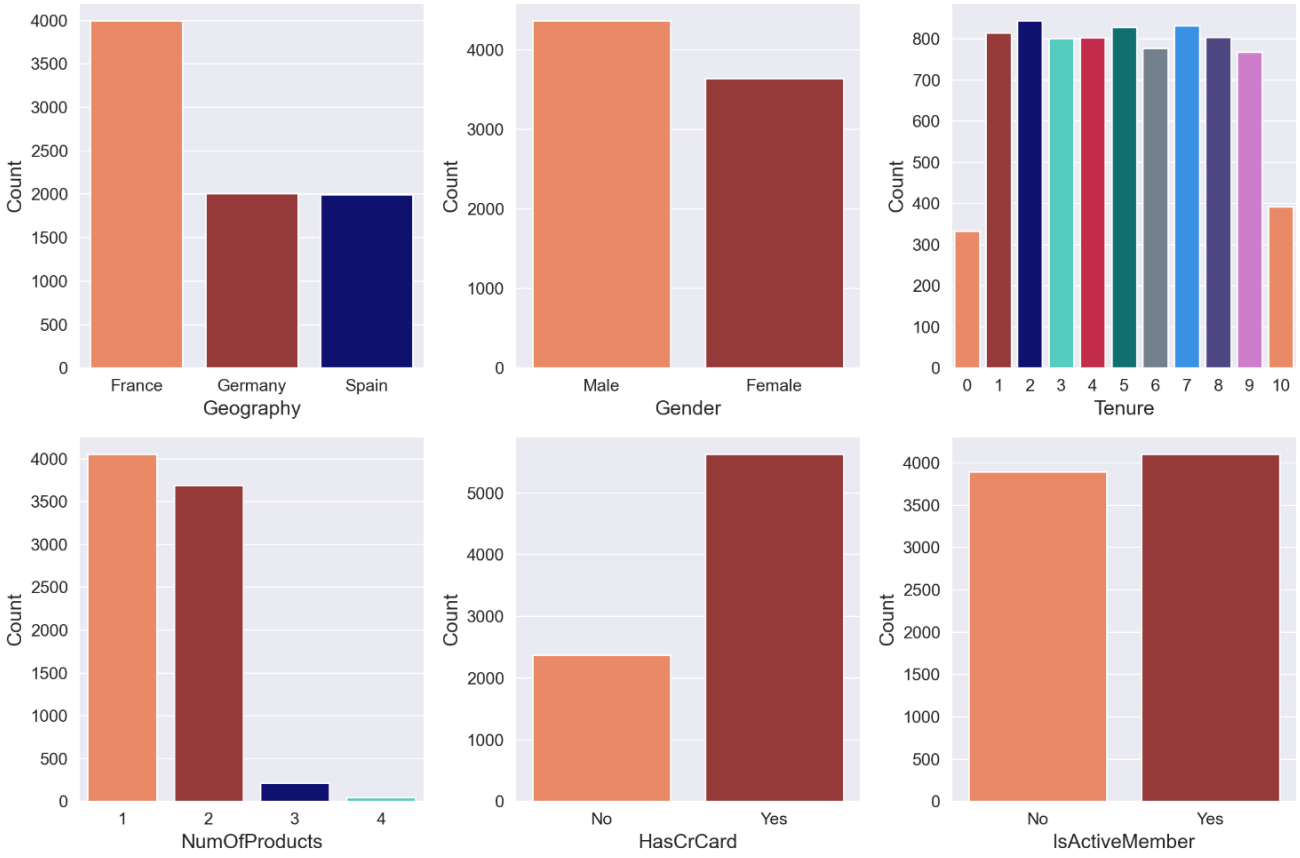


Рис. 3.4. Графіки категоріальних змінних

З цих графіків можна зробити загальні висновки про розподіли інформації за категоріальними змінними, а саме: більшість клієнтів знаходиться у Франції, банк має більше клієнтів чоловічої статі, ніж жіночої, більшість клієнтів має 1 або 2 куплені продукти банку, більшість клієнтів має кредитні картки, та кількість активних та неактивних клієнтів є приблизно рівною. Більш детальний

розгляд по кожній змінній надає додаткову корисну інформацію (рис. 3.5, 3.6 та 3.7).

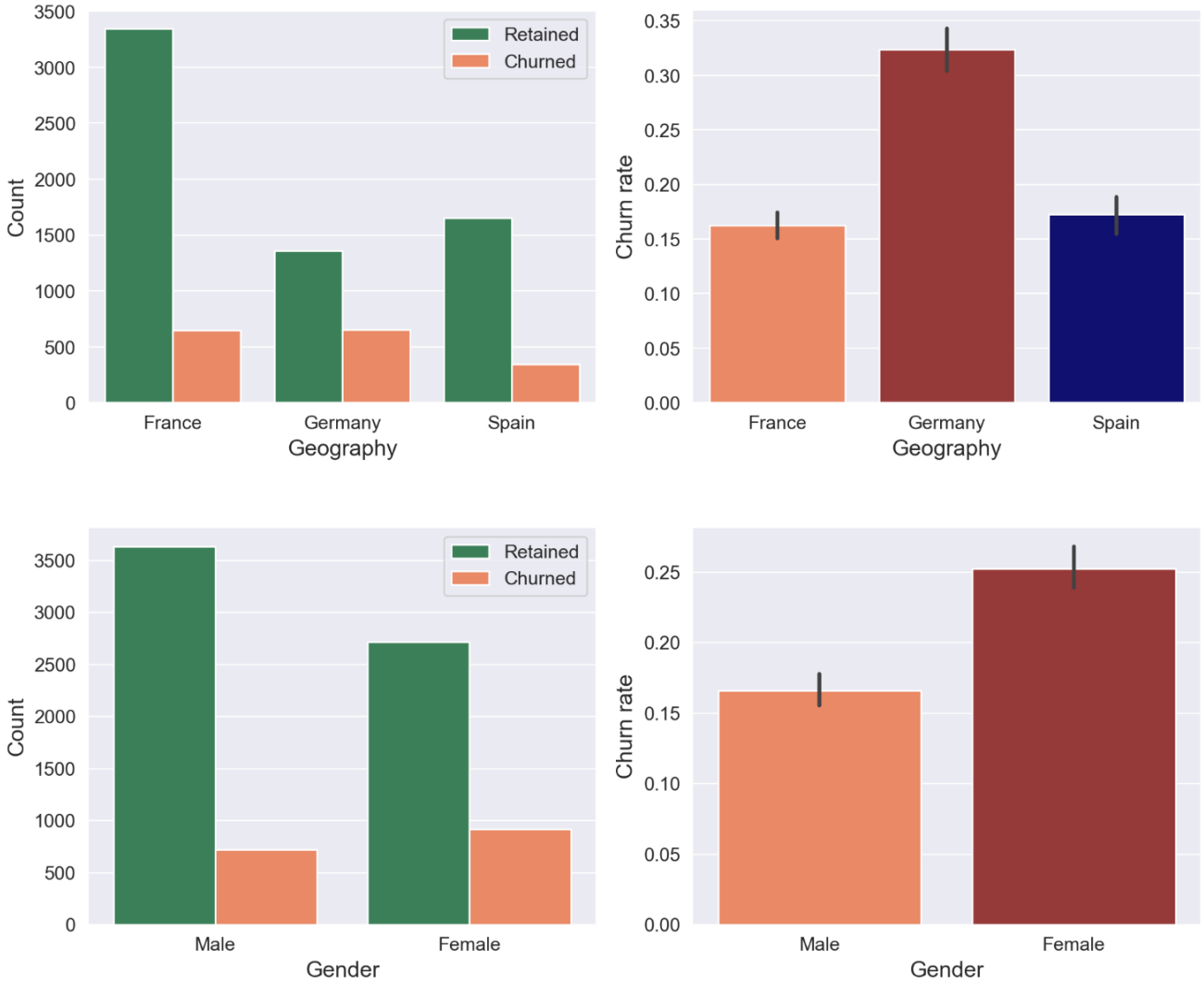


Рис. 3.5. Графіки категоріальних змінних Geography та Gender за цільовою змінною

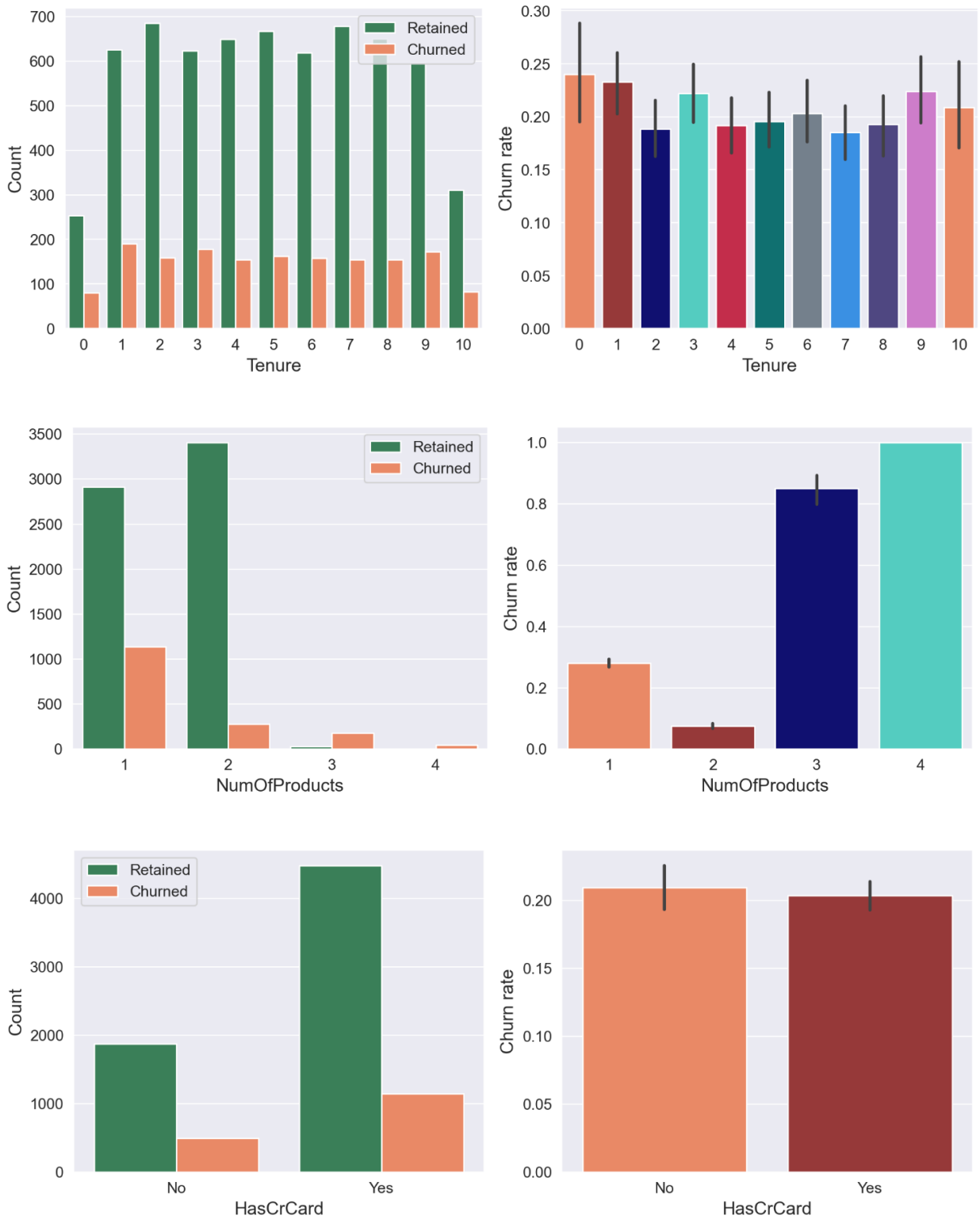


Рис. 3.6. Графіки категоріальних змінних Tenure, NumOfProducts та HasCrCard за цільовою змінною

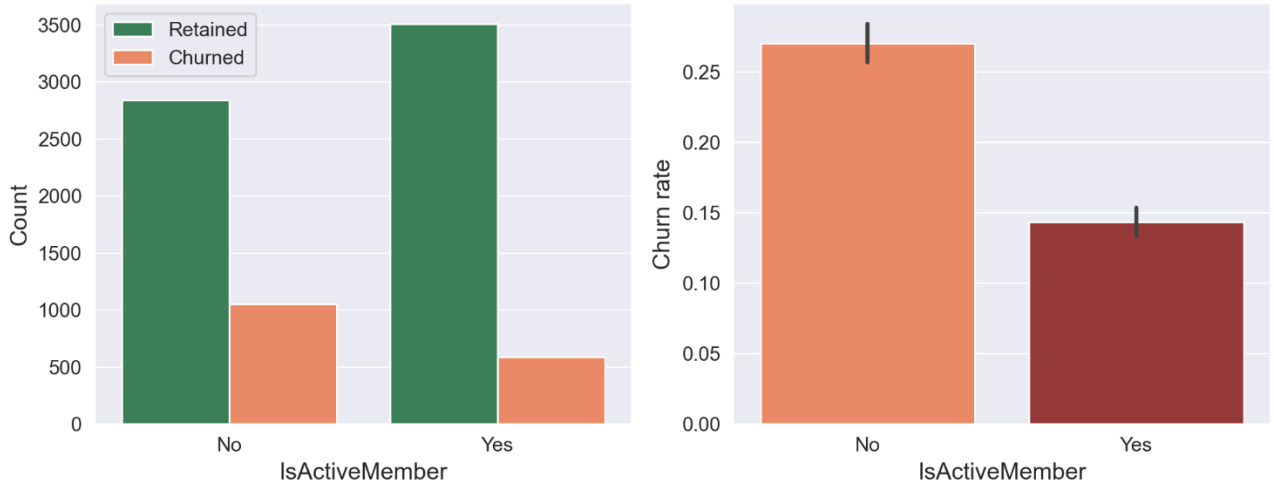


Рис. 3.7. Графіки категоріальної змінної IsActiveMember за цільовою змінною

З цих графіків можна зробити висновки, що відтік клієнтів є активнішим у Німеччині, що може вказувати на локальні проблеми або складнощі з роботою з клієнтами у цьому регіоні. Клієнти-жінки частіше відмовляються від послуг. Наявність у клієнта більше 2 куплених продуктів помітно підвищує шанс відтоку. Це може вказувати на відсутність відповідної підтримки клієнтів з боку банку, що негативно впливає на їх рівень задоволеності. Наявність кредитної картки не має істотного впливу на ймовірність відтоку. І, нарешті, неактивні клієнти мають більший рівень відтоку, що не є дивним.

Після завершення етапу дослідження даних наступним кроком є штучне створення пропусків. Набір даних розділюється на навчальну та тестову вибірки у співвідношенні 80 до 20. Штучні пропуски будуть створюватися лише у навчальній вибірці, оскільки тестова вибірка повинна бути однаковою та повною заради більшої точності під час проведення порівняльної роботи. У роботі було створено ряд наборів даних з різними типами пропусків та різною їх кількістю. Зокрема, було створено 4 набори даних, що мали виключно пропуски типу MCAR:

1. MCAR5v2: має 5% випадкових пропусків за змінними Balance та CreditScore

2. MCAR8v3: має 8% випадкових пропусків за змінними EstimatedSalary, Balance, CreditScore
3. MCAR5: має 5% випадкових пропусків за змінними EstimatedSalary, Balance, Age, CreditScore
4. MCAR15: має 15% випадкових пропусків за змінними EstimatedSalary, Balance, Age, CreditScore

2 набори даних, що мали виключно пропуски типу MAR:

5. MAR1: має пропуски за змінною EstimatedSalary коли змінна Age має значення нижче 20-го перцентиля або вище 90-го. Має пропуски за змінною Balance коли змінна CreditScore має значення нижче 10-го перцентиля або вище 95-го. Має пропуски за змінною CreditScore коли змінна Age має значення вище 90-го перцентиля.
6. MAR3: має пропуски за змінною EstimatedSalary коли змінна Age має значення нижче 7-го перцентиля або вище 93-го. Має пропуски за змінною Balance коли змінна CreditScore має значення нижче 5-го перцентиля або вище 95-го.

2 набори даних, що мають змішані пропуски типу MCAR та MAR:

7. MAR2MCAR: має пропуски за змінною EstimatedSalary коли змінна Age має значення нижче 20-го перцентиля або вище 90-го. Має 10% випадкових пропусків за змінними Balance, Age, CreditScore
8. MAR7MCAR8: має пропуски за змінною EstimatedSalary коли змінна Age має значення нижче 7-го перцентиля або вище 93-го. Має 8% випадкових пропусків за змінними Age, NumOfProducts

1 набір даних, що має пропуски типу MNAR:

9. MNAR7: має пропуски за змінною EstimatedSalary коли змінна вона має значення нижче 7-го перцентиля або вище 93-го. має пропуски за змінною Age коли змінна вона має значення нижче 7-го перцентиля або вище 93-го

Та 3 набори даних, що мають змішані пропуски з використанням пропусків типу MNAR:

10. MNAR7MCAR8: має пропуски за змінною EstimatedSalary коли змінна вона має значення нижче 7-го перцентиля або вище 93-го. має пропуски за змінною Age коли змінна вона має значення нижче 7-го перцентиля або вище 93-го. Має 8% випадкових пропусків за змінними Balance, NumOfProducts
11. MNAR7MAR7: має пропуски за змінною EstimatedSalary коли змінна вона має значення нижче 7-го перцентиля або вище 93-го. має пропуски за змінною Age коли змінна вона має значення нижче 7-го перцентиля або вище 93-го. Має пропуски за змінною Tenure коли змінна Balance має значення вище 90-го перцентиля.
12. MNAR7MAR7MCAR8: має пропуски за змінною EstimatedSalary коли змінна вона має значення нижче 7-го перцентиля або вище 93-го. має пропуски за змінною Age коли змінна вона має значення нижче 7-го перцентиля або вище 93-го. Має пропуски за змінною Tenure коли змінна Balance має значення вище 90-го перцентиля. Має 8% випадкових пропусків за змінними Balance, NumOfProducts

Зазначені набори даних дозволяють перевірити результати роботи методів заповнення різного рівня складності в різних умовах, що не обов’язково відповідатимуть цільовому типу пропусків методу.

Було обрано 4 методи заповнення для дослідження:

1. Аналіз повних спостережень (Listwise Deletion)
2. Заповнення середнім (Mean Imputation)
3. Множинне заповнення Python бібліотеки scikit-learn (Iterative Imputer)
4. Множинне заповнення R бібліотеки MICE

Методи 1 та 2 є класичними, методи 3 та 4 це реалізації алгоритму множинного заповнення від Python бібліотеки scikit-learn та R бібліотеки MICE. У методі 3 клас IterativeImputer моделює кожну ознаку з відсутніми значеннями як функцію інших ознак і використовує цю оцінку для заповнення пропущених даних. Це відбувається ітеративно у циклі "кожен з кожним": на кожному кроці ознака вибирається як цільова (y), а інші ознаки розглядаються як вхідні (X). Для відомих значень у обчислюється регресор (регресійна модель) на (X, y). Потім регресор використовується для прогнозу відсутніх значень y. Це робиться ітеративно для кожної ознаки, а потім повторюється протягом заданої кількості ітерацій. Результати останньої ітерації заповнення повертаються.

У методі 4 кожна змінна має свою модель, що використовується для заповнення. Вбудовані моделі заповнення надаються для неперервних даних (передбачувальне відповідне зіставлення середнього значення), бінарних даних (логістична регресія), невпорядкованих категоріальних даних (політомна логістична регресія) і впорядкованих категоріальних даних (пропорційні шанси). Таким чином, обрано два традиційні методи та два складні: класичний метод обробки пропусків аналізом повних спостережень, базовий метод заповнення середнім та дві реалізації алгоритму множинного заповнення. За допомогою цих методів отримано 4 варіанти заповнених наборів даних для кожного вказаного набору даних: загалом 48 наборів.

У якості алгоритмів прогнозування було обрано наступні 4 алгоритми:

1. Логістична регресія
2. Метод опорних векторів (Support Vector Machines)
3. Метод випадкового лісу (Random Forest Classificator)
4. LGBM (Light Gradient Boosting Machine)

Логістична регресія у пакеті scikit-learn – це реалізація лінійної моделі для розв’язання задачі класифікації з використанням логістичної рівності

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

де $x_0$ – значення $x$, що відповідає середині функції

$L$ – супремум значень функції

$k$ – логістичний коефіцієнт зростання (крутість функції).

На рис. 3.8 наведено графік рівності.

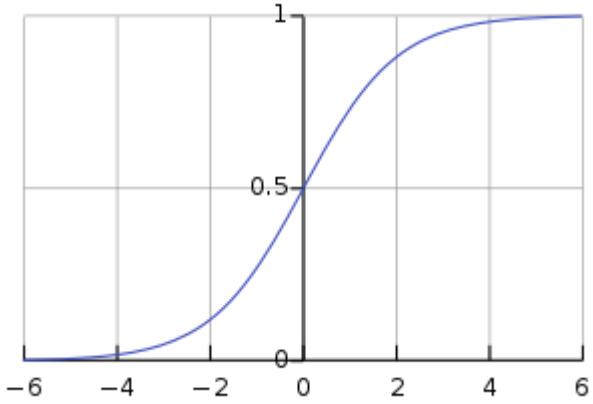


Рис. 3.8. Стандартна логістична рівність, де $L = 1, k = 1, x_0 = 0$

Логістична регресія в scikit-learn підтримує як бінарну, так і багатокласову класифікації.

Метод опорних векторів (SVM) є методом аналізу даних для вирішення завдань класифікації і регресійного аналізу за допомогою моделей з керованим навчанням. В цьому методі використовуються алгоритми навчання, відомі як опорно-векторні машини (SVM), які створюють модель для визначення приналежності нових зразків до одного з двох класів. Модель SVM функціонує як бінарний лінійний класифікатор, який розділяє зразки у просторі, представленому точками, таким чином, щоб між класами існував максимально можливий розрив. Для нових зразків робиться проекція в цей самий простір, і їх класифікація здійснюється на основі того, з якого боку розриву вони розташовані.

Метод випадкового лісу - це алгоритм машинного навчання, який базується на використанні ансамблю дерев, що приймають рішення. Цей алгоритм об'єднує

дві ключові ідеї: метод баггінгу та метод випадкових підпросторів. Метод випадкового лісу застосовується для вирішення завдань класифікації, регресії та кластеризації. Основна концепція полягає в використанні великої кількості рішучих дерев, кожне з яких самостійно дає низьку якість класифікації, але завдяки їх великій кількості, результат стає дуже точним і надійним.

Алгоритм LGBM (Light Gradient Boosting Machine) відноситься до сімейства градієнтного бустингу і використовується для вирішення завдань класифікації, регресії і ранжування. Він починає зі створення простої базової моделі і потім ітеративно додає до неї інші моделі, оптимізуючи їх так, щоб покращити прогнози. LGBM є швидким та оптимізованим алгоритмом, що, серед іншого, має вбудовану підтримку категоріальних ознак без необхідності їх попередньої обробки та надає можливості регуляризації, що допомагає уникнути перенавчання.

Перед початком навчання моделей з кожним створеним набором даних у рамках попередньої обробки були проведені наступні дії:

1. Масштабування даних
2. Обробка незбалансованості набору (семплінг)

Масштабування даних є відіграє важливу роль у забезпеченні успішної роботи моделей. Цей процес полягає в перетворенні значень ознак, щоб вони потрапляли у відповідний діапазон. Масштабування допомагає уникнути впливу великих різниць в масштабах між різними ознаками на роботу алгоритмів. Наприклад, якщо одна ознака має діапазон від 0 до 1, а інша від 0 до 1000, то остання ознака матиме надмірний вплив на результати моделі. Крім того, масштабування допомагає покращити швидкість навчання моделей. Багато алгоритмів, таких як градієнтний спуск, працюють ефективніше, коли дані масштабовані. Великі значення можуть призвести до чисельних переповнень і повільного навчання, тоді як масштабовані дані дозволяють алгоритмам збільшити швидкість

збіжності. Масштабування вирівнює ці різниці і допомагає моделі більш точно оцінити важливість кожної ознаки. Існує кілька методів масштабування, включаючи стандартизацію, нормалізацію та мін-макс масштабування – у роботі використана стандартизація, що перетворює дані, щоб центрувати їх, віднімаючи середнє значення кожної ознаки, а потім масштабує, використовуючи ділення неконстантних ознак на їх стандартне відхилення.

Використання алгоритмів семплінгу виявляється надзвичайно корисним при обробці незбалансованих наборів даних у сфері машинного навчання. Як було виявлено на етапі дослідження даних, клас клієнтів, що відмовились від послуг банку становив лише близько 20% від всіх даних, в той час як в ідеальному випадку необхідно мати рівну кількість даних обох класів. Алгоритми семплінгу допомагають вирішити цю проблему шляхом створення синтетичних прикладів меншого класу на основі існуючих прикладів. У роботі використано алгоритм SMOTE (англ. Synthetic Minority Over-sampling Technique) - він аналізує кожний приклад меншого класу і для кожного прикладу створює синтетичний приклад, який знаходиться в певній відстані від оригінального. Це допомагає збалансувати кількість прикладів у класах і покращує результати моделі. Це особливо важливо в задачах, де правильне розпізнавання меншого класу має велике значення – у обраній задачі класифікації клієнтів найважливішим є максимізація здатності моделі виявляти клієнтів, що з високою ймовірністю відмовляться від послуг. Точність виявлення клієнтів, що залишаться, є менш важливою.

## 3.3 Аналіз результатів роботи

Після завершення попередньої обробки даних було запущено навчання обраних моделей на повному наборі. Як було зазначено раніше, основною метрикою для визначення якості моделі буде recall – оскільки коректне визначення позитивного результати (клієнт відмовиться) є більш важливим за сенсом поставленої задачі. Вважатимемо значення recall 70% мінімальним задовільним результатом. Результати наведені в таблиці 3.1, червоним кольором відмічені найкращі значення за кожною метрикою.

Таблиця 3.1 Результати навчання моделей на повному наборі даних

| | Accuracy | Precision | Recall | AUC |
|---|---|---|---|---|
| **LR** | 0.727502 | 0.727573 | 0.727344 | 0.799392 |
| **SVC** | 0.816945 | 0.820831 | 0.810887 | 0.895356 |
| **RF** | 0.810966 | 0.821958 | 0.793896 | 0.891082 |
| **LGBMC** | 0.819541 | 0.828747 | 0.805538 | 0.900956 |

Бачимо, що на навчальній вибірці кожна модель показує результат вище мінімального. Наступним кроком є перевірка роботи отриманих моделей на тестовій вибірці. Для цього до тестової вибірки застосовуються аналогічні методи попередньої обробки, як і до навчальної. Отримані результати наведені в таблиці 3.2.

Таблиця 3.2 Результати роботи моделей на тестових даних

| FULL | | | |
|---|---|---|---|
| | **Accuracy** | **Precision** | **Recall** |
| **LR** | 0.725000 | 0.386397 | 0.679389 |
| **SVC** | 0.799500 | 0.492982 | 0.715013 |
| **RF** | 0.808500 | 0.508711 | 0.743003 |
| **LGBM** | 0.820500 | 0.530357 | 0.755725 |

Загалом значення recall нижче, ніж було на навчальних даних, що свідчить про перенавчання моделей. Також результати роботи логістичної регресії впали

нижче за обране мінімальне значення 70%. Найкращий отриманий результат становить приблизно 75.6% в моделі LGBM – це означає, що модель здатна успішно виявляти 75.6% клієнтів, що з високою ймовірністю відмовляться від послуг.

Тепер дослідимо роботу моделей на заповнених наборах даних.

1. MCAR5v2

Результати навчання моделей наведені у таблиці 3.3.

Таблиця 3.3 Результати роботи моделей, що були навчені на заповнених даних MCAR5v2

| | MCAR5v2 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.660900 | 0.341346 | 0.729452 |
| | **SVC** | 0.782699 | 0.471939 | 0.633562 |
| | **RF** | 0.790311 | 0.486553 | 0.681507 |
| | **LGBM** | 0.793772 | 0.492500 | 0.674658 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.698750 | 0.376000 | 0.718654 |
| | **SVC** | 0.801250 | 0.509761 | 0.718654 |
| | **RF** | 0.810000 | 0.525386 | 0.727829 |
| | **LGBM** | 0.818125 | 0.541284 | 0.721713 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.667500 | 0.353362 | 0.755352 |
| | **SVC** | 0.796250 | 0.501080 | 0.709480 |
| | **RF** | 0.839375 | 0.596685 | 0.660550 |
| | **LGBM** | 0.838750 | 0.600583 | 0.629969 |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.696875 | 0.374204 | 0.718654 |
| | **SVC** | 0.804375 | 0.515625 | 0.706422 |
| | **RF** | 0.801250 | 0.509847 | 0.712538 |
| | **LGBM** | 0.813750 | 0.533030 | 0.715596 |

Загалом помітний негативний вплив заповнення, оскільки значення метрики recall знизилося в більшості моделей. Несподівано покращилися результати роботи логістичної регресії, хоча й ціною інших метрик – зокрема, з

використанням IterativeImputer отримане значення recall цієї моделі майже досягає рівня моделі LGBM на повному наборі (але інші метрики нижчі, тому ця модель все-таки має гіршу загальну якість). Дуже суттєво знизилися результати алгоритмів на даних, що були отримані шляхом аналізу повних спостережень та даних, що були заповнені з допомогою IterativeImputer. Виключно за метрикою recall найкращою моделлю за цими результатами є логістична регресія, що навчалася на даних, заповнених з IterativeImputer. Однак загалом найкращою моделлю за цими результатами можна вважати моделі RF та LGBM, що навчалися на даних, заповнених середнім.

2. MCAR8v3

Результати навчання моделей наведені у таблиці 3.4.

Таблиця 3.4 Результати роботи моделей, що були навчені на заповнених даних MCAR8v3

| | MCAR8v3 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.663462 | 0.359292 | 0.777778 |
| | **SVC** | 0.787660 | 0.494595 | 0.701149 |
| | **RF** | 0.810897 | 0.534819 | 0.735632 |
| | **LGBM** | 0.798878 | 0.513514 | 0.727969 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.698750 | 0.376000 | 0.718654 |
| | **SVC** | 0.795000 | 0.498911 | 0.700306 |
| | **RF** | 0.809375 | 0.524017 | 0.733945 |
| | **LGBM** | 0.813125 | 0.531818 | 0.715596 |

Продовження таблиці 3.4.

| | | Accuracy | Precision | Recall |
|---|---|---|---|---|
| **Iterative** | **LR** | 0.667500 | 0.353362 | 0.755352 |
| | **SVC** | 0.798125 | 0.504444 | 0.694190 |
| | **RF** | 0.838750 | 0.592992 | 0.672783 |
| | **LGBM** | 0.843750 | 0.611594 | 0.645260 |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.696875 | 0.374204 | 0.718654 |
| | **SVC** | 0.789375 | 0.489130 | 0.688073 |
| | **RF** | 0.806250 | 0.518600 | 0.724771 |
| | **LGBM** | 0.815625 | 0.536364 | 0.721713 |

У цьому випадку помітно кращі результати дав аналіз повних спостережень. Найвищі показники recall знову спостерігаються для логістичної регресії (на аналізі повних спостережень та IterativeImputer). Значення 77.78% по recall поки є найвищим з досягнутих і перевершує кращий результат досягнутий на повному наборі даних. Найбільш збалансованими моделями є LGBM на наборі з алгоритмом MICE (recall 72.2%) та RF на наборі з заповненням середнім (recall 73.4%). Загалом задовільні показники можна побачити на усіх наборах, де було використано заповнення середнім та на майже всіх, де було використано MICE. Результати роботи IterativeImputer знову суттєво нижчі за інші алгоритми.

3. MCAR5

Результати навчання моделей наведені у таблиці 3.5.

Таблиця 3.5 Результати роботи моделей, що були навчені на заповнених даних MCAR5

| | **MCAR5** | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.636015 | 0.344595 | 0.701031 |
| | **SVC** | 0.767050 | 0.482940 | 0.632302 |
| | **RF** | 0.786207 | 0.516216 | 0.656357 |
| | **LGBM** | 0.786973 | 0.516373 | 0.704467 |

Продовження таблиці 3.5.

| | | | | |
|---|---|---|---|---|
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.720625 | 0.394366 | 0.685015 |
| | **SVC** | 0.800625 | 0.508772 | 0.709480 |
| | **RF** | 0.807500 | 0.520971 | 0.721713 |
| | **LGBM** | 0.812500 | 0.531178 | 0.703364 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.685625 | 0.370206 | 0.767584 |
| | **SVC** | 0.813125 | 0.532258 | 0.706422 |
| | **RF** | 0.850625 | 0.618280 | 0.703364 |
| | **LGBM** | 0.852500 | 0.632653 | 0.663609 |
| | | | | |
| **MICE** | | Accuracy | Precision | Recall |
| | **LR** | 0.707500 | 0.385737 | 0.727829 |
| | **SVC** | 0.810625 | 0.527778 | 0.697248 |
| | **RF** | 0.820000 | 0.544622 | 0.727829 |
| | **LGBM** | 0.819375 | 0.543981 | 0.718654 |

Тут результати загалом схожі, але погіршилися метрики моделей, що навчалися на даних з використанням аналізу повних спостережень. При цьому моделі, що навчалися на даних, де використовувався IterativeImputer вперше показали одні з найкращих результатів серед інших методів (за виключенням LGBM). Найвищі показники належать логістичній регресії у поєднанні з IterativeImputer – 76.76%, що знову перевершує кращий результат на повному наборі.

4. MCAR15

Результати навчання моделей наведені у таблиці 3.6.

Таблиця 3.6 Результати роботи моделей, що були навчені на заповнених даних MCAR15

| | MCAR15 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.720238 | 0.375000 | 0.689441 |
| | **SVC** | 0.803571 | 0.491525 | 0.720497 |
| | **RF** | 0.797619 | 0.482072 | 0.751553 |
| | **LGBM** | 0.723810 | 0.389408 | 0.776398 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.713750 | 0.386874 | 0.685015 |
| | **SVC** | 0.787500 | 0.486081 | 0.694190 |
| | **RF** | 0.791875 | 0.493562 | 0.703364 |
| | **LGBM** | 0.793125 | 0.495763 | 0.715596 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.731250 | 0.414023 | 0.758410 |
| | **SVC** | 0.815000 | 0.535632 | 0.712538 |
| | **RF** | 0.845625 | 0.604712 | 0.706422 |
| | **LGBM** | 0.855000 | 0.636888 | 0.675841 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.716875 | 0.396040 | 0.733945 |
| | **SVC** | 0.821875 | 0.549528 | 0.712538 |
| | **RF** | 0.820000 | 0.543046 | 0.752294 |
| | **LGBM** | 0.818750 | 0.542529 | 0.721713 |

Знову помітно гарні результати на результатах з аналізом повних спостережень з кращим результатом для набору даних в моделі LGBM – однак також помітне дуже суттєве зниження значень інших метрик, що до цього жодного разу не відбувалося для цієї моделі. Це показник сильного негативного впливу зниження кількості навчальних даних, що пов’язане з видаленням великої їх частини через наявність великої кількості пропусків у цьому наборі. Також гірше показало себе заповнення середнім. Обидва методи множинного

заповнення добре показали себе в цьому наборі, зокрема, метрики логістичної регресії у поєднанні з реалізацією множинного заповнення від scikit-learn перевершили результати навчання цієї моделі на повному наборі даних. Це можна розглядати як ілюстрацію випадку коли використання заповнених даних може в деяких умовах давати кращі результати, ніж використання повних даних.

5. MAR1

Результати навчання моделей наведені у таблиці 3.7.

Таблиця 3.7 Результати роботи моделей, що були навчені на заповнених даних MAR1

| | MAR1 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.638918 | 0.335784 | 0.643192 |
| | **SVC** | 0.739854 | 0.439344 | 0.629108 |
| | **RF** | 0.772112 | 0.489437 | 0.652582 |
| | **LGBM** | 0.785640 | 0.512915 | 0.652582 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.725000 | 0.398927 | 0.681957 |
| | **SVC** | 0.793750 | 0.496614 | 0.672783 |
| | **RF** | 0.806875 | 0.519565 | 0.730887 |
| | **LGBM** | 0.810625 | 0.526667 | 0.724771 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.667500 | 0.353362 | 0.755352 |
| | **SVC** | 0.807500 | 0.521739 | 0.697248 |
| | **RF** | 0.840000 | 0.601140 | 0.645260 |
| | **LGBM** | 0.843125 | 0.608571 | 0.651376 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.702500 | 0.379254 | 0.715596 |
| | **SVC** | 0.798750 | 0.505695 | 0.678899 |
| | **RF** | 0.798750 | 0.505400 | 0.715596 |
| | **LGBM** | 0.808750 | 0.523596 | 0.712538 |

З введенням пропусків типу MAR одразу стає помітним те, наскільки негативним є вплив аналізу повних спостережень на результати моделей. Метрика recall у всіх моделей впала нижче мінімального задовільного значення 70%. Також незадовільні результати показали моделі, що навчалися на даних, заповнених з IterativeImputer. Найкраще значення recall знову належить логістичній регресії з використанням IterativeImputer. Найбільш якісною моделлю виявилася RF з заповненням середнім, схожий результат на заповненні середнім показала LGBM. Водночас найбільш стабільні результати за всіма моделями можна побачити на результатах навчання за даними з використанням алгоритму MICE.

6. MAR3

Результати навчання моделей наведені у таблиці 3.8.

Таблиця 3.8 Результати роботи моделей, що були навчені на заповнених даних MAR3

| | MAR3 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.716335 | 0.415704 | 0.636042 |
| | **SVC** | 0.772112 | 0.495957 | 0.650177 |
| | **RF** | 0.794422 | 0.533875 | 0.696113 |
| | **LGBM** | 0.807968 | 0.558989 | 0.703180 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.702500 | 0.379254 | 0.715596 |
| | **SVC** | 0.796875 | 0.502262 | 0.678899 |
| | **RF** | 0.813125 | 0.531818 | 0.715596 |
| | **LGBM** | 0.807500 | 0.521158 | 0.715596 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.667500 | 0.353362 | 0.755352 |
| | **SVC** | 0.801875 | 0.511161 | 0.700306 |
| | **RF** | 0.835000 | 0.586301 | 0.654434 |
| | **LGBM** | 0.841250 | 0.603399 | 0.651376 |

Продовження таблиці 3.8.

| | | Accuracy | Precision | Recall |
|---|---|---|---|---|
| MICE | LR | 0.698750 | 0.376000 | 0.718654 |
| | SVC | 0.801875 | 0.511628 | 0.672783 |
| | RF | 0.809375 | 0.525000 | 0.706422 |
| | LGBM | 0.810625 | 0.526549 | 0.727829 |

Загалом результати схожі на попередній випадок. Знову бачимо незадовільні результати на аналізі повних спостережень, хоча модель LGBM змогла досягти мінімально необхідного значення – ймовірно, це спричинено меншою кількістю пропусків. Заповнення середнім показало доволі гарний результат у цьому наборі з трьома моделями задовільного рівня. Кращий результат recall знову належить моделі логістичної регресії на заповненні IterativeImputer. Найбільш якісною моделлю у цьому випадку можна вважати LGBM з MICE та RF з LGBM на заповненні середнім. Варто зазначити, що метрики моделі RF на MICE заповненні з цим набором перевершили всі метрики цієї моделі на повному наборі даних, а recall майже досяг кращого результату за повним набором.

7. MAR2MCAR

Результати навчання моделей наведені у таблиці 3.9.

Таблиця 3.9 Результати роботи моделей, що були навчені на заповнених даних MAR2MCAR

| | MAR2MCAR | | | |
|---|---|---|---|---|
| Listwise | | Accuracy | Precision | Recall |
| | LR | 0.708181 | 0.375862 | 0.652695 |
| | SVC | 0.758242 | 0.435146 | 0.622754 |
| | RF | 0.775336 | 0.466135 | 0.700599 |
| | LGBM | 0.789988 | 0.489451 | 0.694611 |

Продовження таблиці 3.9.

| | | Accuracy | Precision | Recall |
|---|---|---|---|---|
| **Mean** | **LR** | 0.715625 | 0.387324 | 0.672783 |
| | **SVC** | 0.797500 | 0.503311 | 0.697248 |
| | **RF** | 0.801250 | 0.509719 | 0.721713 |
| | **LGBM** | 0.797500 | 0.503240 | 0.712538 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.688125 | 0.374636 | 0.785933 |
| | **SVC** | 0.825625 | 0.555814 | 0.730887 |
| | **RF** | 0.853750 | 0.632479 | 0.678899 |
| | **LGBM** | 0.848750 | 0.619718 | 0.672783 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.698125 | 0.377743 | 0.737003 |
| | **SVC** | 0.808750 | 0.523596 | 0.712538 |
| | **RF** | 0.822500 | 0.548533 | 0.743119 |
| | **LGBM** | 0.818125 | 0.540359 | 0.737003 |

На першому наборі даних з змішаним типом пропусків помітно перевагу методу MICE, що забезпечує прийнятні результати по всім моделям незважаючи на досить велику кількість пропусків. Результат за моделлю RF знову перевершив кращий результат цієї моделі на повному наборі. Стосовно інших методів: IterativeImputer показав себе краще з простішими моделями, а заповнення середнім – з складнішими. Дуже високе значення recall має логістична регресія з IterativeImputer. Аналіз повних спостережень як і в попередніх наборах негативно загалом впливає на роботу моделей.

8. MAR7MCAR8

Результати навчання моделей наведені у таблиці 3.10.

Таблиця 3.10 Результати роботи моделей, що були навчені на заповнених даних MAR7MCAR8

| | MAR7MCAR8 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.659610 | 0.336714 | 0.688797 |
| | **SVC** | 0.779001 | 0.469512 | 0.639004 |
| | **RF** | 0.821338 | 0.551724 | 0.663900 |
| | **LGBM** | 0.810330 | 0.526814 | 0.692946 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.698750 | 0.369748 | 0.672783 |
| | **SVC** | 0.795625 | 0.500000 | 0.703364 |
| | **RF** | 0.805625 | 0.518692 | 0.678899 |
| | **LGBM** | 0.812500 | 0.531616 | 0.694190 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.693125 | 0.377976 | 0.776758 |
| | **SVC** | 0.813750 | 0.532584 | 0.724771 |
| | **RF** | 0.845000 | 0.605333 | 0.694190 |
| | **LGBM** | 0.845625 | 0.608696 | 0.685015 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.707500 | 0.387200 | 0.740061 |
| | **SVC** | 0.813125 | 0.532110 | 0.709480 |
| | **RF** | 0.811875 | 0.527660 | 0.758410 |
| | **LGBM** | 0.813125 | 0.531674 | 0.718654 |

На другому наборі даних з змішаним типом пропусків так само найбільш якісно виглядає метод заповнення MICE. Краще значення recall знову досягнуто логістичною регресією у поєднанні з IterativeImputer. При цьому заповнення середнім цього разу показало себе гірше, ніж у попередніх випадках.

9. MNAR7

Результати навчання моделей наведені у таблиці 3.11.

Таблиця 3.11 Результати роботи моделей, що були навчені на заповнених даних MNAR7

| | MNAR7 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.663651 | 0.355899 | 0.707692 |
| | **SVC** | 0.763158 | 0.461538 | 0.646154 |
| | **RF** | 0.772204 | 0.477089 | 0.680769 |
| | **LGBM** | 0.779605 | 0.488889 | 0.676923 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.743750 | 0.421846 | 0.685015 |
| | **SVC** | 0.783125 | 0.479079 | 0.700306 |
| | **RF** | 0.795000 | 0.498943 | 0.721713 |
| | **LGBM** | 0.797500 | 0.503268 | 0.706422 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.727500 | 0.409917 | 0.758410 |
| | **SVC** | 0.807500 | 0.520879 | 0.724771 |
| | **RF** | 0.841250 | 0.596817 | 0.688073 |
| | **LGBM** | 0.850000 | 0.621170 | 0.681957 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.710000 | 0.389694 | 0.740061 |
| | **SVC** | 0.808125 | 0.521930 | 0.727829 |
| | **RF** | 0.813125 | 0.530568 | 0.743119 |
| | **LGBM** | 0.820000 | 0.545035 | 0.721713 |

Перший набір з наявністю пропусків найскладнішого з точки зору заповнення типу MNAR знову вказує на переваги методу MICE. Теорія методу передбачає необхідність обмеження MAR на пропуски, але серед обраних методів для обраних моделей цей метод показав однозначно найбільш стабільні результати, забезпечивши високі показники по всім моделям, в деяких випадках навіть перевершивши стартові показники (логістична регресія та метод опорних векторів). Також загалом гарні результати показав метод заповнення середнім.

Результати роботи IterativeImputer схожі на попередні де він краще працював у поєднанні з простішими моделями, ніж з складнішими. Аналіз повних спостережень знову доволі негативно впливає на показники моделей.

10.MNAR7MCAR8

Результати навчання моделей наведені у таблиці 3.12.

Таблиця 3.12 Результати роботи моделей, що були навчені на заповнених даних MNAR7MCAR8

| | MNAR7MCAR8 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.651412 | 0.348739 | 0.775701 |
| | **SVC** | 0.761441 | 0.449511 | 0.644860 |
| | **RF** | 0.801363 | 0.517007 | 0.710280 |
| | **LGBM** | 0.810127 | 0.532872 | 0.719626 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.743125 | 0.422222 | 0.697248 |
| | **SVC** | 0.788750 | 0.488069 | 0.688073 |
| | **RF** | 0.790625 | 0.491266 | 0.688073 |
| | **LGBM** | 0.806250 | 0.520581 | 0.657492 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.730000 | 0.412060 | 0.752294 |
| | **SVC** | 0.805625 | 0.517699 | 0.715596 |
| | **RF** | 0.841875 | 0.596859 | 0.697248 |
| | **LGBM** | 0.849375 | 0.618785 | 0.685015 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.716875 | 0.396040 | 0.733945 |
| | **SVC** | 0.796875 | 0.502232 | 0.688073 |
| | **RF** | 0.813125 | 0.530702 | 0.740061 |
| | **LGBM** | 0.818125 | 0.541475 | 0.718654 |

Більш складний механізм утворення пропусків негативно відбився на результатах роботи заповнення середнім, однак показники моделей на аналізі повних спостережень покращились. Результати роботи IterativeImputer та MICE

аналогічні попереднім за виключенням методу опорних векторів на MICE, recall якого суттєво погіршився.

11.MNAR7MAR7

Результати навчання моделей наведені у таблиці 3.13.

Таблиця 3.13 Результати роботи моделей, що були навчені на заповнених даних MNAR7MAR7

| | MNAR7MAR7 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.720037 | 0.410319 | 0.716738 |
| | **SVC** | 0.774016 | 0.478916 | 0.682403 |
| | **RF** | 0.806953 | 0.535714 | 0.708155 |
| | **LGBM** | 0.823422 | 0.566667 | 0.729614 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.740000 | 0.418349 | 0.697248 |
| | **SVC** | 0.780625 | 0.474026 | 0.669725 |
| | **RF** | 0.784375 | 0.481557 | 0.718654 |
| | **LGBM** | 0.813750 | 0.534442 | 0.688073 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.730000 | 0.412060 | 0.752294 |
| | **SVC** | 0.804375 | 0.515217 | 0.724771 |
| | **RF** | 0.840000 | 0.589873 | 0.712538 |
| | **LGBM** | 0.849375 | 0.616848 | 0.694190 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.712500 | 0.391517 | 0.733945 |
| | **SVC** | 0.804375 | 0.515419 | 0.715596 |
| | **RF** | 0.806875 | 0.519313 | 0.740061 |
| | **LGBM** | 0.806250 | 0.518124 | 0.743119 |

Загалом схожі результати на набір даних MNAR7.

12.MNAR7MAR7MCAR8

Результати навчання моделей наведені у таблиці 3.14.

Таблиця 3.14 Результати роботи моделей, що були навчені на заповнених даних MNAR7MAR7MCAR8

| | MNAR7MAR7MCAR8 | | | |
|---|---|---|---|---|
| **Listwise** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.691974 | 0.400000 | 0.717703 |
| | **SVC** | 0.812364 | 0.573770 | 0.669856 |
| | **RF** | 0.837310 | 0.634703 | 0.665072 |
| | **LGBM** | 0.840564 | 0.646226 | 0.655502 |
| | | | | |
| **Mean** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.743750 | 0.423573 | 0.703364 |
| | **SVC** | 0.776875 | 0.467532 | 0.660550 |
| | **RF** | 0.798125 | 0.504587 | 0.672783 |
| | **LGBM** | 0.808125 | 0.524876 | 0.645260 |
| | | | | |
| **Iterative** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.731250 | 0.413445 | 0.752294 |
| | **SVC** | 0.799375 | 0.506726 | 0.691131 |
| | **RF** | 0.843125 | 0.597436 | 0.712538 |
| | **LGBM** | 0.846875 | 0.614525 | 0.672783 |
| | | | | |
| **MICE** | | **Accuracy** | **Precision** | **Recall** |
| | **LR** | 0.713750 | 0.392447 | 0.730887 |
| | **SVC** | 0.796250 | 0.501109 | 0.691131 |
| | **RF** | 0.818750 | 0.542725 | 0.718654 |
| | **LGBM** | 0.811250 | 0.528217 | 0.715596 |

Найбільш змішаний набір даних після заповнення показав загалом схожі результати на попередні, але є відмінності у результатах за моделями SVC та RF – так, SVC цього разу не досяг мінімально необхідного рівня якості з жодним методом заповнення, а RF досяг цього рівня у поєднанні з IterativeImputer, що до цього відбувалося лише на деяких даних з MCAR типом пропусків.

Загалом за всіма проведеними дослідженнями можна зробити наступні висновки:

1. Аналіз повних спостережень, незважаючи на його проблематику з статистичної точки зору, здатний створювати набори даних, навчання на яких може давати результати не гірші, ніж коли використовуються складні методи заповнення. Однак ці випадки є швидше випадковими, ніж постійними, тому надійність методу є доволі сумнівною. Результати роботи методу на даних з MCAR типом пропусків були змішані, на даних з MAR типом пропусків були низькі, а на даних з MNAR та змішаним типом пропусків знову були змішані. Через цю загальну ненадійність можна зробити висновок, що цей метод не є найкращим вибором для вирішення подібних задач.
2. Заповнення середнім загалом показало гарні результати на більшості наборів даних, хоча зазвичай потребувало складних класифікаторів RF або LGBM. Цей метод працював гірше з методом опорних векторів та з логістичною регресією. Найменш якісні результати роботи були отримані на даних з змішаним типом пропусків, що включав MNAR категорію. Загалом цей метод, незважаючи на свою простоту, показав себе з гарної сторони і може використовуватись, якщо необхідно швидко вирішити проблему пропущених даних.
3. Множинне заповнення у реалізації IterativeImputer від Python бібліотеки scikit-learn показало незадовільні результати – моделі, навчені на даних, що були заповнені цим методом, часто не відповідали мінімальній необхідній якості класифікації. Найкраще цей метод працював з логістичною регресією та з методом опорних векторів – зокрема, логістична регресія неодноразово показувала помітно кращі результати по метриці recall при використанні цього заповнення у порівнянні з стартовими повними даними – однак програвала по іншим метрикам, через що загальна якість моделі була нижчою. Моделі RF та LGBM стабільно мали низьке значення recall у

поєднанні з цим заповненням. В той же час варто відмітити, що метрика precision цих моделей у поєднанні з цим заповненням мала суттєво кращі значення, ніж при навчанні на повному наборі даних. Через це можна зробити висновок, що цей складний метод заповнення не відповідав меті обраної задачі, однак може показати себе набагато краще у випадках коли цільовою метрикою є precision. Також можна зазначити, що він є доволі специфічним, оскільки ефект підвищення precision або зниження recall не працює з кожною моделлю класифікації однаково, що ускладнює передбачуваність роботи цього алгоритму заповнення.

4. Множинне заповнення у реалізації від R бібліотеки MICE показало себе з найкращої сторони, забезпечуючи стабільно високі результати по всім метрикам, що були найбільш наближені до показників після навчання на повному наборі даних. Метод працював добре на всіх наборах незалежно від типу, комбінації та кількості пропусків, навіть добре показав себе при роботі з типом пропусків MNAR, до роботи з яким він за теорією не є пристосованим. Найкраще цей метод працював у поєднанні з алгоритмами RF та LGBM. Алгоритм RF у деяких випадках з використанням цього методу заповнення навіть показував кращі результати, ніж після навчання на повному наборі даних. Загалом, цей метод є найбільш надійним вибором серед досліджуваних для обробки пропусків будь-якого роду, хоча його використання може бути пов’язане з деякими додатковими складнощами через його належність до мови R.

## 3.4 Висновки до розділу

В даному розділі розглянуто вибір технологій та описано процес підготовки до проведення дослідження. Створено програмний продукт, що дозволяє легко досліджувати пропущені дані різних типів, методи заповнення, їх вплив на моделі класифікації та прогнозування, порівнювати отримані результати. Було створено дванадцять наборів даних з різними типами, комбінаціями та кількістю пропусків, та проведено аналіз впливу чотирьох методів заповнення на чотири алгоритми класифікації. Найрезультативнішим методом заповнення виявився метод множинного заповнення у реалізації від R бібліотеки MICE.

# РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Завдяки сучасному стрімкому розвитку технологій штучного інтелекту та машинного навчання спостерігається помітне зростання ролі їх використання при створенні стартапів. Це явище свідчить про революційний зсув в підходах до підприємництва та розвитку нових бізнесів. Штучний інтелект дозволяє створювати інноваційні рішення та автоматизувати процеси, що раніше були складними або недосяжними. Машинне навчання допомагає аналізувати великі обсяги даних та прогнозувати тренди, що дозволяє створювати продукти та послуги, які відповідають сучасним вимогам ринку. Зростання ролі штучного інтелекту у сучасних стартапах нерозривно пов'язане з необхідністю наявності якісних навчальних даних та використання методів заповнення пропусків. Спостерігається зростання обсягів даних, які використовуються для тренування інтелектуальних систем, і водночас збільшення кількості пропущених значень в цих даних. Важливість надійних та повних даних стає критичною, оскільки якість навчальних даних безпосередньо впливає на якість моделей штучного інтелекту.

Методи заповнення пропущених даних дозволяють ефективно використовувати навчальні дані, навіть якщо вони не є повністю заповненими. Це робить можливим навчання надійних моделей машинного навчання навіть на даних із пропусками. У цьому розділі розглянуто процес розробки стартап-проекту системи дослідження впливу використання методів заповнення пропущених даних на якість роботи прогнозних моделей машинного навчання.

## 4.1 Опис ідеї проекту

Методи заповнення пропущених даних є невід'ємною частиною обробки і аналізу даних у сучасному машинному навчанні та аналітиці. Проте, важливо розуміти, що ці методи не є універсальними і необхідно враховувати численні специфічні фактори, які впливають на їхню ефективність та статистичну точність. Пропущені дані можуть бути в результаті різних причин, включаючи технічні помилки, відсутність відомої інформації або навіть випадковий характер. Тип пропуску може вимагати використання специфічних методів заповнення, таких як інтерполяція часового ряду, видалення аномальних значень або використання класифікації для заповнення категоріальних даних. Також при невеликій кількості пропущених значень можуть використовуватися прості методи, такі як середнє значення або медіана. Проте, в разі великої кількості пропусків, бажано враховувати більш складні стратегії, включаючи використання алгоритмів машинного навчання для прогнозування пропущених значень на основі інших ознак. Ці факти підкреслюють, що вибір оптимального методу заповнення пропущених даних є актуальною проблемою. Ідеєю стартапу є вибір найкращого методу заповнення, що максимізує результативність обраної моделі прогнозування. Опис ідеї подано у таблиці 4.1.

Таблиця 4.1 Опис ідеї стартап-проекту

| Зміст ідеї | Напрямки застосування | Вигоди для користувача |
|---|---|---|
| Система надає користувачам можливість завантажувати свої набори даних і моделі прогнозування. Система проводить аналіз даних та моделей, визначає типи пропусків та відповідність методів заповнення, а потім рекомендує оптимальний метод для максимізації точності моделі. | Не обмежені жодною конкретною галуззю чи групою даних. Він може бути використаний в різних галузях, таких як фінанси, медицина, маркетинг, наукові дослідження та інші, де важлива точність прогнозів. | Система допомагає користувачам знайти оптимальний метод заповнення, що призводить до покращення точності прогнозування та робить моделі надійнішими. Підходить для різних галузей та завдань прогнозування, роблячи її корисною для різних професій та сфер діяльності. |

Аналіз техніко-економічних переваг ідеї, що включає аналіз конкурентних проектів подано у таблиці 4.2.

Таблиця 4.2 Визначення сильних, слабких та нейтральних характеристик ідеї проекту

| № п/п | Техніко-економічні характеристики ідеї | Потенційні товари/концепції конкурентів | | | W (слабка сторона) | N (нейтральна сторона) | S (сильна сторона) |
|---|---|---|---|---|---|---|---|
| | | Мій проект | Autoimpute | Q | | | |
| 1 | Вартість ПЗ | Низька | Низька | Висока | | | + |
| 2 | Доступність | Висока | Висока | Середня | | | + |
| 3 | Підтримка | + | + | + | | + | |
| 4 | Простота у використанні | + | + | Власна система, що вимагає попередньої підготовки користувача | | | + |

З таблиці 4.2 можна зробити висновок про конкурентоспроможність розробленої системи.

## 4.2 Технологічний аудит проекту

Технологічну можливість реалізувати ідею проекту описано в таблиці 4.3.

Таблиця 4.3 Технологічна здійсненність ідеї проекту

| № п/п | Ідея проекту | Технологія реалізації | Наявність технологій | Доступність технологій |
|---|---|---|---|---|
| 1 | Створення системи для автоматизації вибору методу заповнення | Програмні пакети Python для обробки та аналізу даних, побудови моделей прогнозування, за потреби можуть використовуватися програмні пакети мови R. | Наявні | У вільному доступі |
| Обрані технології реалізації ідеї проекту: мови програмування Python, R, програмні пакети реалізацій необхідних методів заповнення та прогнозування, дистрибутив Anaconda | | | | |

Результати аналізу говорять про те, що технологічна реалізація проекту можлива. Для цього будуть використані можливості мов програмування Python та R, що найбільше підходять для обраної задачі та здатні надати весь необхідний функціонал для її реалізації.

## 4.3 Аналіз ринкових можливостей запуску стартап-проекту

У першу чергу необхідно провести аналіз попиту на продукт. Він наведений в таблиці 4.4.

Таблиця 4.4 Попередня характеристика потенційного ринку стартап-проекту

| № п/п | Показники стану ринку (найменування) | Характеристика |
|---|---|---|
| 1. | Кількість головних гравців, од | |
| 2. | Загальний обсяг продаж, грн/ум.од. | |
| 3. | Динаміка ринку (якісна оцінка) | Зростає |
| 4. | Наявність обмежень для входу (вказати характер обмежень) | Немає |
| 5. | Специфічні вимоги до стандартизації та сертифікації | Немає |
| 6. | Середня норма рентабельності в галузі (або по ринку), % | |

У таблиці 4.5 наведено характеристику клієнтів та їх вимог до продукту.

Таблиця 4.5 Характеристика потенційних клієнтів стартап-проекту

| Потреба, що формує ринок | Цільова аудиторія (цільові сегменти ринку) | Відмінності у поведінці різних потенційних цільових груп клієнтів | Вимоги споживачів до товару |
|---|---|---|---|
| Наявність якісних даних для навчання алгоритмів штучного інтелекту та машинного навчання | Дослідники та розробники ПЗ, що працюють з технологіями штучного інтелекту, бізнес-аналітики, менеджери проектів, корпорації, стартапи | Цільові групи клієнтів мають різний бюджет та пріоритети, різні очікування від результатів | Можливості інтеграції продукту у сторонні платформи аналізу даних, наявність якісної візуалізації результатів, точність результатів, конфіденційність даних, швидкість та простота у користуванні, наявність специфічних методів заповнення |

Наступним етапом є дослідження ринкового середовища, факторів та загроз, а також можливостей.

Таблиця 4.6 Фактори загроз

| № п/п | Фактор | Зміст загрози | Можлива реакція компанії |
|---|---|---|---|
| 1 | Конкуренція великих гравців на ринку | Великі корпорації та вже усталені гравці на ринку можуть випускати конкурентні продукти або купувати конкуруючі стартапи. | Необхідно зосередитися на інноваціях та надавати перевагу своїм конкурентним перевагам, таким як точність аналізу чи зручність використання. |
| 2 | Технологічна вразливість | Швидкі зміни в технологіях та безпеці можуть створити вразливості у програмному забезпеченні, які можуть бути використані зловмисниками. | Необхідно активно вдосконалювати системи безпеки, регулярно оновлювати програмне забезпечення та реагувати на виявлені вразливості. |
| 3 | Зміни у споживчому попиті | Зміни в споживчому попиті, такі як зміна популярності певних методів аналізу даних, можуть вплинути на попит на продукт. | Необхідно мати гнучку стратегію та здатність адаптуватися до змін в споживчому попиті. Це може включати розвиток нових функцій чи зміну рекламної стратегії. |
| 4 | Загрози щодо конфіденційності даних | Порушення конфіденційності даних клієнтів може призвести до втрати довіри і репутації. | Варто приділяти велику увагу захисту даних, використовувати сучасні методи шифрування та створювати міцні політики конфіденційності. |

Розбір факторів можливостей наведено у таблиці 4.7.

Таблиця 4.7 Фактори можливостей

| № п/п | Фактор | Зміст можливості | Можлива реакція компанії |
|---|---|---|---|
| 1 | Зростаючий попит на аналіз даних | Зростаючий попит на аналіз даних вказує на постійну потребу в інструментах та послугах, пов'язаних з аналізом та оптимізацією даних. | Вдосконалення свого продукту та надавати рішення, які відповідають специфічним потребам клієнтів |
| 2 | Зростання кількості даних | За зростанням обсягів даних створюється потреба в ефективніших методах обробки та аналізу. | Розробка рішення для оптимізації обробки великих обсягів даних, використовуючи розподілені обчислення та інші технології. |
| 3 | Розвиток IoT | З IoT з'являється багато нових джерел даних, які можна аналізувати. | Інтеграція з системами IoT та надавати інструменти для аналізу даних, отриманих з сенсорів та IoT-пристроїв. |
| 4 | Міжнародний ринок | Аналітика даних є глобальною потребою | Розширення присутності на міжнародному ринку, локалізація продукту та надання підтримки мовами різних регіонів. |

Надалі необхідно провести дослідження пропозиції та оцінити загальні характеристики конкуренції на ринку. Ступеневий аналіз конкуренції наведено у таблиці 4.8.

Таблиця 4.8 Ступеневий аналіз конкуренції на ринку

| Особливості конкурентного середовища | В чому проявляється дана характеристика | Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною) |
|---|---|---|
| 1. Вказати тип конкуренції - досконала. | Незалежні компанії та продукти, що конкурують між собою на рівних умовах. | Необхідно пропонувати конкурентоспроможний продукт, що зможе відповідати потребам широкого кола користувачів. |
| 2.За рівнем конкурентної боротьби - міжнародний. | Компанії розташовані по всьому світу. | Співпраця з іноземними партнерами, необхідність приділення значної уваги доступності продукту за кордоном, робота над локалізаціями та їх підтримкою. |
| 3.За галузевою ознакою - внутрішньогалузева. | Послуги пропонуються компаніями у галузі аналізу даних. | Регулярне вдосконалення функціоналу, розширення підтримки типів оброблюваних даних. |

Продовження таблиці 4.8

| | | |
|---|---|---|
| 4.Конкуренція за видами товарів: товарно-видова | Конкуренція між видами систем аналізу даних, використанням методів заповнення даних, їх особливостями та специфікацією. | Розроблена система повинна бути максимально гнучким продуктом, що здатний задовільнити потреби клієнтів краще за конкурентів |
| 5.За характером конкурентних переваг – нецінова | Вдосконалення систем автоматизації обробки пропущених даних | Розрахунок ресурсів з урахуванням витрат на дослідження та експерименти з новими методами, моделями та типами даних |
| 6.За інтенсивністю: не марочна | Бренд практично не грає ролі. | Реклама продукту, маркетингова робота з поширення інформації. |

Аналіз конкуренції в галузі за моделлю М. Портера було виконано в таблиці 4.9.

Таблиця 4.9 Аналіз конкуренції в галузі за М. Портером

| Складові аналізу | Прямі конкуренти в галузі | Потенційні конкуренти | Постачальники | Клієнти | Товари-замінники |
|---|---|---|---|---|---|
| | Навести перелік прямих конкурентів | Визначити бар'єри входження в ринок | Визначити фактори сили постачальників | Визначити фактори сили споживачів | Фактори загроз з боку замінників |
| Висновки: | Визначено 2 конкуренти на ринку, найбільш схожа реалізація ідея виконана в проекті autoimpute – це ПЗ у формі бібліотеки, що знаходиться у вільному доступі | Можливості входу на ринок присутні через наявність покращеної системи визначення кращого методу. | Фактори сили постачальників відсутні. | Для користувачів важливими є кількість підтримуваних методів та моделей, якість виконаного аналізу. | Товари-замінники можуть використати якіснішу технологію визначення методів, що здатна зменшити цінність нашого товару. |

Отже, вихід на ринок є доцільним. Наступним кроком є визначення переліку факторів конкурентоспроможності розробленої системи на основі аналізів, проведених у таблицях 4.2, 4.5, 4.6, 4.7 та 4.9. Цей перелік наведено у таблиці 4.10.

Таблиця 4.10 Обґрунтування факторів конкурентоспроможності

| № п/п | Фактор конкурентоспроможності | Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим) |
|---|---|---|
| 1. | Функціонал програмного продукту | Програмний продукт має повністю задовольняти потреби користувачів та характеризуватися якістю виконання своїх задач. Алгоритми, методи та моделі, що підтримуються у продукті, повинні постійно вдосконалюватись та розширюватись. |
| 2 | Ціноутворення та доступність | Продукт повинен бути доступним різним користувачам з різним рівнем та масштабом потреб. |
| 3 | Гнучкість та налаштованість | Можливість пристосовувати продукт до специфічних потреб клієнтів та галузевих особливостей робить його універсальним та привабливим для різних секторів. |
| 4 | Інтеграція з іншими інструментами | Можливість інтеграцій в існуючі популярні продукти з аналізу даних є привабливим фактором, що полегшує процеси використання продукту. |

На основі проведеного аналізу факторів конкурентоспроможності визначаються сильні та слабкі сторони стартап-проекту (таблиця 4.11).

Таблиця 4.11 Порівняльний аналіз сильних та слабких сторін проекту

| № п/п | Фактор конкурентоспроможності | Бали 1-20 | Рейтинг товарів-конкурентів у порівнянні з розробленим продуктом | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| 1 | Функціонал програмного продукту | 15 | | | | | + | | |
| 2 | Ціноутворення та доступність | 20 | | | | | + | | |
| 3 | Гнучкість та налаштованість | 20 | | | | + | | | |
| 4 | Інтеграція з іншими інструментами | 10 | | | | | | + | |

Фінальним етапом ринкового аналізу можливостей стартап-проекту є SWOT-аналіз на основі всіх описаних сильних та слабких сторін, конкуренції, загроз та можливостей. SWOT-аналіз наведено у таблиці 4.12.

Таблиця 4.12 SWOT-аналіз стартап-проекту

| | |
|---|---|
| Сильні сторони: технологічна інноваційність, автоматизація, продуктивність, гнучкість, налаштованість | Слабкі сторони: конкуренція, залежність від технологічних змін |
| Можливості: зростаючий попит на аналіз даних, розширення застосувань в галузях, зростання кількості даних, розвиток IoT | Загрози: конкуренція великих учасників ринку, технологічна вразливість, зміни у споживчому попиті, загрози конфіденційності |

За результатами SWOT-аналізу визначено альтернативи ринкової стратегії, що дозволить вивести розроблену систему як стартап на ринок (таблиця 4.13).

Таблиця 4.13 Альтернативи ринкового впровадження стартап-проекту

| Альтернатива (орієнтовний комплекс заходів) ринкової поведінки | Ймовірність отримання ресурсів | Строки реалізації |
|---|---|---|
| Вихід на ринок із програмним продуктом з розширеним функціоналом | 85% | 6 місяців |
| Вихід на ринок із програмним продуктом, що має розширений функціонал та підтримку інтеграцій з поширеними системами аналізу даних | 70% | 10 місяців |

## 4.4 Розроблення ринкової стратегії проекту

Етап розробки ринкової стратегії проекту в першу чергу передбачає визначення стратегії охоплення ринку (таблиця 4.14).

Таблиця 4.14 Вибір цільових груп потенційних споживачів

| Опис профілю цільової групи потенційних клієнтів | Готовність споживачів сприйняти продукт | Орієнтовний попит в межах цільової групи (сегменту) | Інтенсивність конкуренції в сегменті | Простота входу у сегмент |
|---|---|---|---|---|
| Бізнес-сектор (компанії) | Середній, оскільки великі компанії зазвичай використовують зарекомендовані продукти інших великих компаній | Середній | Висока | Висока складність, вимагає високого рівня компетенції та інвестицій у дослідження та розробку |
| Наукові та дослідницькі установи | Висока, оскільки дослідницькі установи активно використовують різноманітні способи аналізу даних у своїх проектах | Високий | Середня-висока | Середня-висока складність, вимагає здатності створювати спеціалізовані рішення для досліджень |

Продовження таблиці 4.14.

| | | | | |
|---|---|---|---|---|
| Стартапи та малі бізнеси | Висока, оскільки стартапи часто шукають шляхи для оптимізації та вдосконалення своїх операцій і для них важлива невисока вартість продукту | Високий | Середня, але має тенденцію до зростання | Середня-висока, вимагає здатності відповідати потребам стартапів та малих бізнесів |

Найбільш перспективною цільовою групою є третя група. Для роботи у обраному сегменті ринку потрібно мати стратегію розвитку (таблиця 4.15)

Таблиця 4.15 Визначення базової стратегії розвитку

| Обрана альтернатива розвитку проекту | Стратегія охоплення ринку | Ключові конкурентоспроможні позиції відповідно до обраної альтернативи | Базова стратегія розвитку |
|---|---|---|---|
| Вихід на ринок із програмним продуктом з розширеним функціоналом | Визначення та задоволення потреб цільового сегмента | Регулярні оновлення програмного продукту, забезпечення стабільного процесу роботи, поліпшення якості результатів | Стратегія диференціації |

Обґрунтування вибору стратегії диференціації наведено в таблиці 4.16.

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

| Чи є проект «першопрохідцем» на ринку? | Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів? | Чи буде компанія копіювати основні характеристики товару конкурента, і які? | Стратегія конкурентної поведінки |
|---|---|---|---|
| Ні | Будуть здійснюватися обидва варіанти | Може запозичувати окремі базові концепції | Стратегія зайняття конкурентної ніші |

На основі обраної базової стратегії та вимог споживачів формуються шляхи розроблення стратегії позиціонування (таблиця 4.17)

Таблиця 4.17 - Визначення стратегії позиціонування

| Вимоги до товару цільової аудиторії | Базова стратегія розвитку | Ключові конкурентоспроможні позиції власного стартап- проекту | Вибір асоціацій, які мають сформувати позицію власного проекту |
|---|---|---|---|
| Точність результатів, варіативність підтримуваних методів та моделей, підтримка різних типів даних, доступність, легкість інтеграцій | Стратегія диференціації | Функціонал, точність вибору, практичність результатів, просте використання | Точність вибору методу, якість аналізу |

## 4.5 Розроблення маркетингової програми стартап-проекту

Формування маркетингової концепції товару є першим кроком розробки маркетингової програми проекту, воно наведено у таблиці 4.18.

Таблиця 4.18 - Визначення ключових переваг концепції потенційного товару

| Потреба | Вигода, яку пропонує товар | Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити |
|---|---|---|
| Потреба в оптимізації процесу аналізу даних. | Зменшення часу та зусиль, потрібних для аналізу та обробки даних. | Використання передових методів машинного навчання, що дозволяють отримувати точніші результати |
| Візуалізація результатів | Якісна, детальна та зрозуміла візуалізація процесів роботи та результатів | Забезпечення високої точності та інформативності візуалізації, що використовує інтуїтивний інтерфейс |
| Потреба в інтеграції з існуючими інструментами для аналізу даних. | Можливість легко інтегрувати продукт з інструментами аналізу даних з використанням Python чи R. | Зручні інтерфейси для створення інтеграцій |
| Конфіденційність та безпека даних | Високий рівень захисту та конфіденційності даних користувачів. | Використання передових методів шифрування та забезпечення безпеки. |

Трирівнева маркетингова модель допомагає легше розуміти, як продукт еволюціонує від ідеї до реалізації та успішного позиціонування на ринку (таблиця 4.19).

Таблиця 4.19 - Опис трьох рівнів моделі товару

| Рівні товару | Сутність та складові |
|---|---|
| I. Товар за задумом | Інструмент для аналізу даних та заповнення пропущених даних, що базується на передових алгоритмах машинного навчання. |
| II. Товар у реальному виконанні | Програмне забезпечення або хмарна платформа, яку користувачі можуть використовувати для аналізу даних та заповнення пропущених даних. |
| III. Товар із підкріпленням | Регулярні оновлення та покращення ПЗ, підтримка користувачів, навчальні програми |

Наступним етапом є визначення цінових меж, що допомагають з встановленням ціни на потенційний товар (таблиця 4.20).

Таблиця 4.20 - Визначення меж встановлення ціни

| Рівень цін на товари-замінники | Рівень цін на товари-аналоги | Рівень доходів цільової групи споживачів | Верхня та нижня межі встановлення ціни на товар/послугу |
|---|---|---|---|
| - | Вільний доступ або 20-200$/місяць | 1000$/місяць | 10-50$/місяць |

Таблиця 4.21 містить визначення оптимальної системи збуту.

Таблиця 4.21 - Формування системи збуту

| Специфіка закупівельної поведінки цільових клієнтів | Функції збуту, які має виконувати постачальник товару | Глибина каналу збуту | Оптимальна система збуту |
|---|---|---|---|
| Стартапи та малі бізнеси: чутливі до цін, обмежені фінансовими ресурсами. Обмежена закупівля зі змінами в залежності від потреб | Продаж, підтримка клієнтів | Нульовий та перший рівні | Багатоканальні системи |

Концепція маркетингових комунікацій наводиться у таблиці 4.22.

Таблиця 4.22 - Концепція маркетингових комунікацій

| Специфіка поведінки цільових клієнтів | Канали комунікацій, якими користуються цільові клієнти | Ключові позиції, обрані для позиціонування | Завдання рекламного повідомлення | Концепція рекламного звернення |
|---|---|---|---|---|
| Стартапи та малі бізнеси: обмежена закупівля зі змінами за потреби | Веб-сайти, електронна пошта, соціальні мережі, пошукові системи | Точність роботи, якість результату, швидкість, простота у використанні | Наголос на тому, що продукт здатний задовільнити потреби клієнта | Швидка та якісна допомога у аналізі даних для навчання прогнозних моделей |

## 4.6 Висновки до розділу

Внаслідок проведення аналізу створеної системи як стартап-проекту було зроблено висновок, що можливість виходу на ринок з даним проектом є реальною, адже ринок характеризується високим попитом і подібні продукти мають гарну рентабельність.

Ключовими конкурентоспроможними особливостями проекту є здатність точно визначати оптимальний метод заповнення для кожного конкретного випадку пропущених даних, наявність якісною та деталізованої візуалізації даних, інтуїтивний інтерфейс та невисока вартість.

# ВИСНОВКИ

У дисертації було розглянуто основні способи роботи з пропущеними даними, зокрема в наборах даних, що використовуються у навчанні прогнозних моделей машинного навчання, оскільки в реальному світі дані часто є неповними і неідеальними, а для отримання прогнозів, що матимуть високу точність, необхідним є використання якісних навчальних даних. Різні методи заповнення пропущених даних демонструють різний рівень ефективності в залежності від контексту та характеру даних. На це, зокрема, впливають такі фактори як характер утворення пропусків або їх кількість.

У роботі було розглянути як найпростіші традиційні методи обробки пропущених даних, серед яких аналіз повних спостережень, заповнення середнім, медіаною, інтерполяційні методи – так і складні методи, що враховують контекст та взаємозв'язки між даними – наприклад, множинне заповнення. Загалом, складні методи демонструють кращі результати, оскільки ставлять за мету збереження статистичних параметрів даних, але можуть бути більш витратними з точки зору обчислювальних ресурсів, часу та простоти використання у порівнянні з традиційними методами.

Було реалізовано систему для проведення дослідження впливу методів заповнення на різні прогнозні моделі – вона складається з етапів дослідження та обробки вхідних даних, модуля заповнення даних, створення та навчання прогнозних моделей та візуалізації результатів. В роботі були досліджені методи аналізу повних спостережень, заповнення середнім та дві реалізації методу множинного заповнення. У якості об'єкту досліджень було взято набір даних банківських клієнтів та задачу прогнозування відтоку клієнтів. Найкращі результати були досягнуті з використанням реалізації R MICE алгоритму множинного заповнення.

Для подальших досліджень можна виділити наступні напрямки:

1. дослідження більшої кількості моделей та методів;
2. дослідження на інших задачах прогнозування;
3. автоматизація процесів визначення кращого методу заповнення.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. Craig K. Enders. Applied Missing Data Analysis: The Guilford Press; 1 ed., 2010. 20 p.
2. Donald B. Rubin. Inference and Missing Data: Biometrika Vol. 63, No. 3. 1976. P. 581-592.
3. A Novel Missing Data Imputation Approach for Time Series Air Quality Data Based on Logistic Regression: Mei Chen, Hongyu Zhu, Yongxu Chen et. al.; MDPI, 2022, №1. P. 3-20.
4. Літтл Р. Дж. А. Рубін Д.Б. Статистичний аналіз даних з пропусками: Фінанси та статистика, 1991. 336 с.
5. Загоруйко Н. Г. Прикладні методи аналізу даних та знань: Н. Г. Загоруйко. – Новосибірськ: Видавництво інституту математики, 1991. 278 с.
6. Steffen Moritz. Thomas Bartz-Beielstein. imputeTS: Time Series Missing Value Imputation in R. The R Journal, №9/1, 2017. P. 207-218.
7. Степашко И. С. Ю. В. Коппа, Г. О. Іутинська. Метод відновлення пропущених даних в екологічних задачах на основі МГУА. URL: http://www.gmdh.net/articles/rus/gaps.pdf.
8. Карлов И. А. Восстановление пропущенных данных при численном моделировании сложных динамических систем. Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление, 2013, 6 (186). С. 137-144.
9. Плотников Д. Е., Миклашевич Т. С., Барталёв С. А. Восстановление временных рядов данных дистанционных измерений методом полиномиальной аппроксимации в скользящем окне переменного размера.

Современные проблемы дистанционного зондирования Земли из космоса, 2014, Т. 11, № 2. С. 103-110.

10. James Honaker, Gary King, Matthew Blackwell. AMELIA II: a program for missing data / Journal of Statistical Software, 2011, Vol. 45, iss. 7, URL: https://cran.r-project.org/web/packages/Amelia/vignettes/amelia.pdf.
11. Gabriel L. Schlomer, Sheri Bauman, Noel A. Card. Best practices for missing data management in counseling psychology / Journal of Counseling Psychology, 2010, Vol. 57, No. 1. P. 1-10.
12. Soley-Bori M. Dealing with missing data: key assumptions and methods for applied analysis. Technical Report, 2013, No. 4. P. 1-20.
13. Susianto Y., Notodiputro K. A., Kurnia A., Wijayanto H. A comparative study of imputation methods for estimation of missing values of per capita expenditure in central java. IOP Conf. Series: Earth and Environmental Science, 2017, 58. P. 1-10.
14. Sung-Suk Chung, Young-Min Chun, Sun-Kyung Lee. A Study on imputation using adjusted cohen method. Journal of the Korean Data & Information Science Society, 2006, Vol. 17, No. 3. P. 871-888.
15. Allison P. D. Missing data. URL: https://statisticalhorizons.com/wp-content/uploads/2012/01/Milsap-Allison.pdf.
16. Therese D. Pigott. A review of methods for missing data. Educational Research and Evaluation, 2001, Vol. 7, No. 4. P. 353-383.
17. Бочаров Б. П., Воеводина М. Ю. Анализ эффективности алгоритма восстановления пропущенных значений временного ряда результатов тестирования знаний. Системи обробки інформації, 2008, Вип. 3(70). С. 1-13.

18. Кутлалиев А. Х. Метод множественного восстановления данных. Социологические методы в современной исследовательской практике: сб. ст., посвященных памяти А. О. Крыштановского, М., 2011. С. 201-207. URL: http://www.isras.ru/files/File/Sociologicheskie_methody.pdf.
19. Снитюк В. Е. Эволюционный метод восстановления пропусков в данных: сборник трудов VI Международной конференции “Интеллектуальный анализ информации”. Київ, 2006. С. 262-271. URL: http://masters.donntu.org/2012/iii/shkarpetkina/library/article2.html
20. Абраменкова И. В., Круг В. В. Методы восстановления пропусков в массивах данных. Программные продукты и системы, 2005, № 2. С. 18-22.
21. Злоба Е., Яцкив И. Статистические методы восстановления пропущенных данных. Computer Modelling & New Technologies, 2002, Vol. 6, No. 1. P. 51-61.
22. Волошко А. В., Бедерак Я. С., Лутчин Т. Н., Кудрицкий М. Ю. К вопросу восстановления учетных данных на химических предприятиях. Известия Томского политехнического университета, 2014, Т. 324, № 5. С. 101-106.
23. Радчикова Е. С. Анализ применения способов заполнения пропусков в данных во временных рядах в экологических исследованиях. Экология и защита окружающей среды: сб. тез. докл. Междунар. науч.-практ. конф., 19–20 марта 2014 г., Минск, 2014. С. 112-116.
24. Калинин А. В., Ченцов С. В. Алгоритм восстановления пропусков на поле “плохих” данных. Сибирский журнал науки и технологий – основное научное издание Сибирского гос. университета науки и технологий им. акад. М. Ф. Решетнева, 2008, Т. 18, № 2. С. 91-95.
25. Попов А.Ю., Макаренко О.С., Бідюк П.І. Розв’язання задачі заповнення пропусків даних альтернативними методами при створенні прогнозних моделей. ІІ Всеукраїнська науково-практична конференція «Системні

науки та інформатика», 4–8 грудня 2023 р., Київ : КПІ ім. Ігоря Сікорського, 2023. С. 201-206.

# ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

**1. helperFunctions.py**

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import numpy as np
import pandas as pd
pd.set_option('display.precision', 3)

# Data Visualisation Libraries
import matplotlib.pyplot as plt
# %config InlineBackend.figure_format = 'retina'

# !pip install seaborn --upgrade
import seaborn as sns
sns.set_style('darkgrid')

# Statistics
from scipy.stats import chi2_contingency
from imblearn.over_sampling import SMOTE

# Machine Learning
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import learning_curve

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
VotingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

from sklearn.metrics import accuracy_score, recall_score, precision_score, auc,
roc_auc_score, roc_curve
from sklearn.metrics import confusion_matrix
import scikitplot as skplt

font_size = 13
plt.rcParams['axes.labelsize'] = font_size
plt.rcParams['axes.titlesize'] = font_size + 2
plt.rcParams['xtick.labelsize'] = font_size - 2
plt.rcParams['ytick.labelsize'] = font_size - 2
plt.rcParams['legend.fontsize'] = font_size - 2
```

```
def plot_conf_mx(cm, ax):
    '''Plot a confusion matrix in the specified axes object.'''
    sns.heatmap(data=cm,
                annot=True,
                cmap='BuPu',
                annot_kws={'fontsize': 20},
                ax=ax)

    ax.set_xlabel('Predicted Label')
    ax.set_xticks([0.5, 1.5])
    ax.set_xticklabels(['Retained', 'Churned'])

    ax.set_ylabel('True Label')
    ax.set_yticks([0.25, 1.25])
    ax.set_yticklabels(['Retained', 'Churned']);


def plot_learning_curve(estimator,
                        X,
                        y,
                        ax,
                        cv=None,
                        train_sizes=np.linspace(0.1, 1.0, 5)):
    '''Plot the learning curves for an estimator in the specified axes
object.'''
    train_sizes, train_scores, test_scores = learning_curve(
        estimator,
        X,
        y,
        cv=cv,
        n_jobs=-1,
        train_sizes=train_sizes,
        scoring='accuracy')

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    ax.fill_between(train_sizes,
                    train_scores_mean - train_scores_std,
                    train_scores_mean + train_scores_std,
                    alpha=0.1,
                    color='dodgerblue')
    ax.fill_between(train_sizes,
                    test_scores_mean - test_scores_std,
                    test_scores_mean + test_scores_std,
                    alpha=0.1,
                    color='darkorange')
```

```
    ax.plot(train_sizes,
            train_scores_mean,
            color='dodgerblue',
            marker='o',
            linestyle='-',
            label='Training Score')
    ax.plot(train_sizes,
            test_scores_mean,
            color='darkorange',
            marker='o',
            linestyle='-',
            label='Cross-validation Score')

    ax.set_xlabel('Training Examples')
    ax.set_ylabel('Score')
    ax.legend(loc='best', fontsize=14);
```

**2. addMissingDataHelper.py**

```
## REQUIRED LIBRARIES
# For data wrangling
import numpy as np
import pandas as pd
import random

# For visualization
import matplotlib.pyplot as plt
# %matplotlib inline
import seaborn as sns
pd.options.display.max_rows = None
pd.options.display.max_columns = None

def removeDataMCAR(df, column_name, percentage):
    if percentage < 0 or percentage > 1:
        raise ValueError("Percentage must be between 0 and 1")

    num_rows_to_remove = int(len(df) * percentage)

    rows_to_remove = random.sample(df.index.tolist(), num_rows_to_remove)

    df.loc[rows_to_remove, column_name] = None

    return df.copy()

def removeDataArrayMCAR(df, column_names, percentages):
    if any(percentage < 0 or percentage > 1 for percentage in percentages):
        raise ValueError("Percentages must be between 0 and 1")

    if len(column_names) != len(percentages):
        raise ValueError("Length of column_names and percentages must match")
```

```
    modified_df = df.copy()

    for column_name, percentage in zip(column_names, percentages):
        num_rows_to_remove = int(len(df) * percentage)

        rows_to_remove = random.sample(df.index.tolist(), num_rows_to_remove)

        modified_df.loc[rows_to_remove, column_name] = None

    return modified_df.copy()

def removeDataMARBelow(df, columnName, columnDependencyName, percentage):
    threshold = df[columnDependencyName].quantile(percentage / 100)

    df.loc[df[columnDependencyName] < threshold, columnName] = None

    return df

def removeDataMARAbove(df, columnName, columnDependencyName, percentage):
    threshold = df[columnDependencyName].quantile(percentage / 100)
    df.loc[df[columnDependencyName] > threshold, columnName] = None
    return df

def removeDataMNARBelow(df, column_name, percentile):
    threshold = df[column_name].quantile(percentile / 100)

    df[column_name] = np.where(df[column_name] < threshold, np.nan,
df[column_name])

    return df

def removeDataMNARAbove(df, column_name, percentile):
    threshold = df[column_name].quantile(percentile / 100)

    df[column_name] = np.where(df[column_name] > threshold, np.nan,
df[column_name])

    return df



import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import numpy as np
import pandas as pd
pd.set_option('display.precision', 3)
```

**3. Full dataset EDA and modelling**

```
# Data Visualisation Libraries
```

```
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

!pip install seaborn --upgrade
import seaborn as sns
sns.set_style('darkgrid')

# Statistics
from scipy.stats import chi2_contingency
from imblearn.over_sampling import SMOTE

# Machine Learning
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import learning_curve

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
VotingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

from sklearn.metrics import accuracy_score, recall_score, precision_score, auc,
roc_auc_score, roc_curve
from sklearn.metrics import confusion_matrix
import scikitplot as skplt

font_size = 13
plt.rcParams['axes.labelsize'] = font_size
plt.rcParams['axes.titlesize'] = font_size + 2
plt.rcParams['xtick.labelsize'] = font_size - 2
plt.rcParams['ytick.labelsize'] = font_size - 2
plt.rcParams['legend.fontsize'] = font_size - 2

colors = ['seagreen', 'coral']
colors_cat = ['coral', 'brown', 'navy', 'turquoise', 'crimson', 'teal',
'slategrey', 'dodgerblue', 'darkslateblue', 'orchid']
colors_comp = ['coral', 'brown', 'navy', 'turquoise', 'crimson', 'teal',
'slategrey']

random_state = 42
scoring_metric = 'recall'
comparison_dict, comparison_test_dict = {}, {}

print('Defaults set')

def plot_continuous(feature):
```

```
    '''Plot a histogram and boxplot for the churned and retained distributions
for the specified feature.'''
    df_func = train_df.copy()
    df_func['Exited'] = df_func['Exited'].astype('category')

    fig, (ax1, ax2) = plt.subplots(2,
                                   figsize=(6, 4),
                                   sharex=True,
                                   gridspec_kw={'height_ratios': (.7, .3)})

    for df, color, label in zip([df_retained, df_churned], colors, ['Retained',
'Churned']):
        sns.histplot(data=df,
                     x=feature,
                     bins=15,
                     color=color,
                     alpha=0.66,
                     edgecolor='firebrick',
                     label=label,
                     kde=False,
                     ax=ax1)
    ax1.legend()

    sns.boxplot(x=feature, y='Exited', data=df_func, palette=colors, ax=ax2)
    ax2.set_ylabel('')
    ax2.set_yticklabels(['Retained', 'Churned'])

    plt.tight_layout();

def plot_categorical(feature):
    '''For a categorical feature, plot a seaborn.countplot for the total counts
of each category next to a barplot for the churn rate.'''
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))

    sns.countplot(x=feature,
                  hue='Exited',
                  data=train_df,
                  palette=colors,
                  ax=ax1)
    ax1.set_ylabel('Count')
    ax1.legend(labels=['Retained', 'Churned'])

    sns.barplot(x=feature,
                y='Exited',
                data=train_df,
                palette=colors_cat,
                ax=ax2)
    ax2.set_ylabel('Churn rate')

    if (feature == 'HasCrCard' or feature == 'IsActiveMember'):
        ax1.set_xticklabels(['No', 'Yes'])
```

```
        ax2.set_xticklabels(['No', 'Yes'])

    plt.tight_layout();

def clf_performance(classifier, classifier_name, classifier_name_abv):
    '''Display the overall performance of a classifier with this template.'''
    print('\n', classifier_name)
    print('------------------------------')
    print('   Best Score ({}): '.format(scoring_metric) +
str(np.round(classifier.best_score_, 3)))
    print('   Best Parameters: ')
    for key, value in classifier.best_params_.items():
        print('      {}: {}'.format(key, value))

    y_pred_pp = cross_val_predict(estimator=classifier.best_estimator_,
                                  X=X_train,
                                  y=y_train,
                                  cv=5,
                                  method='predict_proba')[:, 1]
    y_pred = y_pred_pp.round()

    cm = confusion_matrix(y_train, y_pred, normalize='true')

    fpr, tpr, _ = roc_curve(y_train, y_pred_pp)
    comparison_dict[classifier_name_abv] = [
        accuracy_score(y_train, y_pred),
        precision_score(y_train, y_pred),
        recall_score(y_train, y_pred),
        roc_auc_score(y_train, y_pred_pp), fpr, tpr
    ]

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9, 4))

    plot_conf_mx(cm, ax1)
    plot_learning_curve(classifier.best_estimator_, X_train, y_train, ax2)

    plt.tight_layout();

def plot_feature_imp(classifier, classifier_name, color, ax):
    '''Plot the importance of features for a classifier as a barplot.'''
    importances = pd.DataFrame({'Feature': X_train.columns,
                                'Importance':
np.round(classifier.best_estimator_.feature_importances_, 3)})

    importances = importances.sort_values('Importance',
ascending=True).set_index('Feature')

    importances.plot.barh(color=color,
                          edgecolor='firebrick',
                          legend=False,
                          ax=ax)
```

```
    ax.set_title(classifier_name)
    ax.set_xlabel('Importance');

def test_func(classifier, classifier_name, ax):
    '''Assess the performance on the test set and plot the confusion matrix.'''
    y_pred = classifier.predict(X_test)
    cm = confusion_matrix(y_test, y_pred, normalize='true')

    comparison_test_dict[classifier_name] = [accuracy_score(y_test, y_pred),
                                             precision_score(y_test, y_pred),
                                             recall_score(y_test, y_pred)]

    sns.heatmap(cm,
                annot=True,
                annot_kws={'fontsize': 24},
                cmap='BuPu',
                ax=ax)

    ax.set_title(classifier_name)

    ax.set_xlabel('Predicted Label')
    ax.set_xticks([0.5, 1.5])
    ax.set_xticklabels(['Retained', 'Churned'])

    ax.set_ylabel('True Label')
    ax.set_yticks([0.2, 1.4])
    ax.set_yticklabels(['Retained', 'Churned']);

from helperFunctions import plot_conf_mx, plot_learning_curve

df = pd.read_csv('./1-datasets/Churn_Modelling.csv')

print(df.shape)
df.head()

df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)
df.columns

df.info()

df.describe()

train_df, test_df = train_test_split(df, test_size=0.2,
random_state=random_state)

train_df.reset_index(drop=True, inplace=True)
test_df.reset_index(drop=True, inplace=True)

print('Train set: {} rows x {} columns'.format(train_df.shape[0],
                                               train_df.shape[1]))
print(' Test set: {} rows x {} columns'.format(test_df.shape[0],
```

```
                                                test_df.shape[1]))

fig, ax = plt.subplots(figsize=(4, 4))

sns.countplot(x='Exited', data=train_df, palette=colors, ax=ax)

for index, value in enumerate(train_df['Exited'].value_counts()):
    label = '{}%'.format(round((value / train_df['Exited'].shape[0]) * 100, 2))
    ax.annotate(label,
                xy=(index, value + 250),
                ha='center',
                va='center',
                color=colors[index],
                size=font_size)

ax.set_xticklabels(['Retained', 'Churned'])
ax.set_xlabel('Status')
ax.set_ylabel('Count')
ax.set_ylim([0, 7000]);

continuous = ['Age', 'CreditScore', 'Balance', 'EstimatedSalary']
categorical = ['Geography', 'Gender', 'Tenure', 'NumOfProducts', 'HasCrCard',
'IsActiveMember']

print('Continuous: ', ', '.join(continuous))
print('Categorical: ', ', '.join(categorical))

train_df[continuous].hist(figsize=(8, 6),
                          bins=20,
                          layout=(2, 2),
                          color='lightsalmon',
                          edgecolor='seagreen',);

fig, ax = plt.subplots(figsize=(5, 4))

sns.heatmap(train_df[continuous].corr(),
            annot=True,
            annot_kws={'fontsize': 12},
            cmap='BuPu',
            ax=ax)

ax.tick_params(axis='x', rotation=45)
ax.tick_params(axis='y', rotation=360);

df_churned = train_df[train_df['Exited'] == 1]
df_retained = train_df[train_df['Exited'] == 0]

plot_continuous('Age')

plot_continuous('CreditScore')
```

```
plot_continuous('Balance')

plot_continuous('EstimatedSalary')

df_cat = train_df[categorical]

fig, ax = plt.subplots(2, 3, figsize=(12, 8))

for index, column in enumerate(df_cat.columns):

    plt.subplot(2, 3, index + 1)
    sns.countplot(x=column, data=train_df, palette=colors_cat)

    plt.ylabel('Count')
    if (column == 'HasCrCard' or column == 'IsActiveMember'):
        plt.xticks([0, 1], ['No', 'Yes'])

plt.tight_layout();

plot_categorical('Geography')

plot_categorical('Gender')

plot_categorical('Tenure')

plot_categorical('NumOfProducts')

plot_categorical('HasCrCard')

plot_categorical('IsActiveMember')

train_df['Gender'] = LabelEncoder().fit_transform(train_df['Gender'])

train_df['Geography'] = train_df['Geography'].map({
    'Germany': 1,
    'Spain': 0,
    'France': 0
})

print('Features Encoded!')

file_path = '2_full_eda_df_train_befScaling.csv'

# Write the DataFrame to a CSV file
train_df.to_csv(file_path, index=False)

# Further modification and modeling
### Scaling
scaler = StandardScaler()

scl_columns = ['CreditScore', 'Age', 'Balance',
```

```
               'EstimatedSalary'
              ]
train_df[scl_columns] = scaler.fit_transform(train_df[scl_columns])

file_path = '2_full_eda_df_train_afterScaling.csv'

# Write the DataFrame to a CSV file
train_df.to_csv(file_path, index=False)

y_train = train_df['Exited']
X_train = train_df.drop('Exited', 1)

### Sampling
y_train.value_counts()
over = SMOTE(sampling_strategy='auto', random_state=random_state)
X_train, y_train = over.fit_resample(X_train, y_train)

y_train.value_counts()

### Baseline Models

clf_list = [('Gaussian Naive Bayes', GaussianNB()),
            ('Logistic Regression',
LogisticRegression(random_state=random_state))]

cv_base_mean, cv_std = [], []
for clf in clf_list:

    cv = cross_val_score(estimator=clf[1],
                         X=X_train,
                         y=y_train,
                         scoring=scoring_metric,
                         cv=5,
                         n_jobs=-1)

    cv_base_mean.append(cv.mean())
    cv_std.append(cv.std())

print('Baseline Models (Recall):')

for i in range(len(clf_list)):
    print('   {}: {}'.format(clf_list[i][0], np.round(cv_base_mean[i], 2)))

### Models
lr = LogisticRegression(random_state=random_state)

param_grid = {
    'max_iter': [100],
    'penalty': ['l1', 'l2'],
    'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
    'solver': ['lbfgs', 'liblinear']
```

```
}

lr_clf = GridSearchCV(estimator=lr,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_lr_clf = lr_clf.fit(X_train, y_train)
clf_performance(best_lr_clf, 'Logistic Regression', 'LR')

svc = SVC(probability=True, random_state=random_state)

param_grid = tuned_parameters = [{'kernel': ['rbf'],
                                  'gamma': ['scale', 'auto'],
                                  'C': [.1, 1, 2]},
                                 {'kernel': ['linear'],
                                  'C': [.1, 1, 10]}
                                ]

svc_clf = GridSearchCV(estimator=svc,
                       param_grid=param_grid,
                       scoring=scoring_metric,
                       cv=5,
                       verbose=False,
                       n_jobs=-1)

best_svc_clf = svc_clf.fit(X_train, y_train)
clf_performance(best_svc_clf, 'Support Vector Classifier', 'SVC')

rf = RandomForestClassifier(random_state=random_state)
param_grid = {
    'n_estimators': [100],
    'criterion': ['entropy', 'gini'],
    'bootstrap': [True, False],
    'max_depth': [6],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [2, 3, 5],
    'min_samples_split': [2, 3, 5]
}

rf_clf = GridSearchCV(estimator=rf,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_rf_clf = rf_clf.fit(X_train, y_train)
clf_performance(best_rf_clf, 'Random Forest', 'RF')
```

```
lgbmc = LGBMClassifier(random_state=random_state)

param_grid = {
    'max_depth': [5],
    'num_leaves': [5, 10],
    'learning_rate': [0.001, 0.01],
    'n_estimators': [200],
    'feature_fraction': [0.5],
    'min_child_samples': [5, 10],
    'reg_alpha': [0.1, 0.5],
    'reg_lambda': [0.1, 0.5]
}

lgbmc_clf = GridSearchCV(estimator=lgbmc,
                         param_grid=param_grid,
                         scoring=scoring_metric,
                         cv=5,
                         verbose=False,
                         n_jobs=-1)

best_lgbmc_clf = lgbmc_clf.fit(X_train, y_train)
clf_performance(best_lgbmc_clf, 'LGBMClassifier', 'LGBMC')

### Feature importance
colors_fi = ['steelblue', 'darkgray', 'cadetblue', 'bisque']

fig = plt.subplots(1, 1, figsize=(12, 10))

for i, (name, clf) in enumerate(zip(['RF', 'GB', 'XGB', 'LGBM'],
                                    [best_rf_clf,
#                                       best_gbc_clf,
#                                       best_xgb_clf,
                                     best_lgbmc_clf])):

    ax = plt.subplot(2, 2, i + 1)
    plot_feature_imp(clf, name, colors_fi[i], ax)
    plt.ylabel('')

plt.tight_layout();

### Performance comparison
comparison_matrix = {}
for key, value in comparison_dict.items():
    comparison_matrix[str(key)] = value[0:4]

comparison_df = pd.DataFrame(comparison_matrix,
                             index=['Accuracy', 'Precision', 'Recall', 'AUC']).T
comparison_df.style.highlight_max(color='indianred', axis=0)

comparison_df.plot(kind='bar',
```

```
                  figsize=(10, 5),
                  fontsize=12,
                  color=['darkslateblue', 'dimgray', 'crimson',
'midnightblue'])

plt.legend(loc='upper center',
           fontsize=font_size - 6,
           ncol=len(comparison_df.columns),
           bbox_to_anchor=(0.5, 1.12))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.73, s='0.70', size=font_size + 2, color='red');

fig, ax = plt.subplots(figsize=(10, 5))

for index, key in enumerate(comparison_dict.keys()):
    auc, fpr, tpr = comparison_dict[key][3], comparison_dict[key][4],
comparison_dict[key][5]
    ax.plot(fpr,
            tpr,
            color=colors_comp[index],
            label='{}: {}'.format(key, np.round(auc, 3)))

ax.plot([0, 1], [0, 1], 'k--', label='Baseline')

ax.set_title('ROC Curve')
ax.set_xlabel('True Positive Rate')
ax.set_xticks([0, 0.25, 0.5, 0.75, 1])
ax.set_ylabel('False Positive Rate')
ax.set_yticks([0, 0.25, 0.5, 0.75, 1])
ax.autoscale(axis='both', tight=True)
ax.legend(fontsize=14);

### Test set
# test_df = test_df.drop(features_drop, axis=1)

test_df['Gender'] = LabelEncoder().fit_transform(test_df['Gender'])
test_df['Geography'] = test_df['Geography'].map({
    'Germany': 1,
    'Spain': 0,
    'France': 0
})

test_df[scl_columns] = scaler.transform(test_df[scl_columns])  # not
fit_transform, scaler has already been trained

y_test = test_df['Exited']
X_test = test_df.drop('Exited', 1)
```

```
print('Preprocessing Complete!')

# tuned_voting_soft.fit(X_train, y_train)

fig, ax = plt.subplots(4, 1, figsize=(5, 15))

for i, (name, clf) in enumerate(zip(['LR', 'SVC', 'RF',
                                     'LGBM',
                                    ],
                                    [best_lr_clf.best_estimator_,
                                     best_svc_clf.best_estimator_,
                                     best_rf_clf.best_estimator_,
                                     best_lgbmc_clf.best_estimator_,
                                    ])):
    test_func(clf, name, ax=ax[i])

plt.tight_layout();

comparison_test_df = pd.DataFrame(comparison_test_dict,
                                  index=['Accuracy', 'Precision', 'Recall']).T
comparison_test_df.style.highlight_max(color='indianred', axis=0)

comparison_test_df.plot(kind='bar',
                        figsize=(10, 5),
                        fontsize=12,
                        color=['darkslateblue', 'dimgray', 'crimson'])

plt.legend(loc='upper center',
           ncol=len(comparison_test_df.columns),
           bbox_to_anchor=(0.5, 1.11))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.72, s='0.70', size=font_size + 2, color='red');
```

**4. Adding missing data and creating imputed datasets**

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import numpy as np
import pandas as pd
pd.set_option('display.precision', 3)

# Data Visualisation Libraries
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

!pip install seaborn --upgrade
```

```
import seaborn as sns
sns.set_style('darkgrid')

# Statistics
from scipy.stats import chi2_contingency
from imblearn.over_sampling import SMOTE

# Machine Learning
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import learning_curve

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
VotingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

from sklearn.metrics import accuracy_score, recall_score, precision_score, auc,
roc_auc_score, roc_curve
from sklearn.metrics import confusion_matrix
import scikitplot as skplt

from helperFunctions import  plot_conf_mx, plot_learning_curve
from addMissingDataHelper import removeDataMCAR, removeDataArrayMCAR,
removeDataMARBelow, removeDataMARAbove, removeDataMNARBelow, removeDataMNARAbove

### Read the full dataset (before scaling) + make missing MCAR5

df_train = pd.read_csv('./2-progress-csvs/2_full_eda_df_train_befScaling.csv')

print('Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df_train.shape[0],
df_train.shape[1]))
df_train.head()

df_train.isnull().sum()

df_train_MCAR5 = removeDataArrayMCAR(df_train,
                                          ['EstimatedSalary', 'Balance', 'Age',
'CreditScore'],
                                          [0.05, 0.05, 0.05, 0.05])
# Check columns list and missing values
df_train_MCAR5.isnull().sum()

file_path = '2_df_train_MCAR5.csv'
```

```
# Write the DataFrame to a CSV file
df_train_MCAR5.to_csv(file_path, index=False)

### Read the full dataset (before scaling) + make missing MCAR15

df_train = pd.read_csv('./2-progress-csvs/2_full_eda_df_train_befScaling.csv')

print('Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df_train.shape[0],
df_train.shape[1]))
df_train.head()

df_train.isnull().sum()

df_train_MCAR40 = removeDataArrayMCAR(df_train,
                                          ['EstimatedSalary', 'Balance', 'Age',
'CreditScore'],
                                          [0.15, 0.15, 0.15, 0.15])
# Check columns list and missing values
df_train_MCAR40.isnull().sum()

file_path = '2_df_train_MCAR15.csv'

# Write the DataFrame to a CSV file
df_train_MCAR40.to_csv(file_path, index=False)

### Read the full dataset (before scaling) + make missing MAR1

df_train = pd.read_csv('./2-progress-csvs/2_full_eda_df_train_befScaling.csv')

print('Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df_train.shape[0],
df_train.shape[1]))
df_train.head()

df_train.isnull().sum()

df_train_MAR1 = removeDataMARBelow(df_train, 'EstimatedSalary', 'Age', 20)
df_train_MAR1 = removeDataMARAbove(df_train_MAR1, 'EstimatedSalary', 'Age', 90)
df_train_MAR1 = removeDataMARBelow(df_train_MAR1, 'Balance', 'CreditScore', 10)
df_train_MAR1 = removeDataMARAbove(df_train_MAR1, 'Balance', 'CreditScore', 95)
df_train_MAR1 = removeDataMARAbove(df_train_MAR1, 'CreditScore', 'Age', 90)

# Check columns list and missing values
df_train_MAR1.isnull().sum()

file_path = '2_df_train_MAR1.csv'

# Write the DataFrame to a CSV file
df_train_MAR1.to_csv(file_path, index=False)
```

```
### Read the full dataset (before scaling) + make missing MAR2MCAR

df_train = pd.read_csv('./2-progress-csvs/2_full_eda_df_train_befScaling.csv')

print('Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df_train.shape[0],
df_train.shape[1]))
df_train.head()

df_train.isnull().sum()

df_train_MAR2MCAR = removeDataMARBelow(df_train, 'EstimatedSalary', 'Age', 20)
df_train_MAR2MCAR = removeDataMARAbove(df_train_MAR2MCAR, 'EstimatedSalary',
'Age', 90)
df_train_MAR2MCAR = removeDataArrayMCAR(df_train_MAR2MCAR,
                                        ['Balance', 'Age', 'CreditScore'],
                                        [0.1, 0.1, 0.1])

# Check columns list and missing values
df_train_MAR2MCAR.isnull().sum()

file_path = '2_df_train_MAR2MCAR.csv'

# Write the DataFrame to a CSV file
df_train_MAR2MCAR.to_csv(file_path, index=False)

… (випущено) …

## Imputation Phase
### 1. Listwise Deletion

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MCAR5.csv')

df_missing.isnull().sum()

df_missing.shape

df_listwise = df_missing.dropna()
df_listwise.shape

file_path = '2_df_train_MCAR5_listwise.csv'

# Write the DataFrame to a CSV file
df_listwise.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MCAR15.csv')

df_missing.isnull().sum()

df_listwise = df_missing.dropna()
df_listwise.shape
```

```
file_path = '2_df_train_MCAR15_listwise.csv'

# Write the DataFrame to a CSV file
df_listwise.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MAR1.csv')

df_missing.isnull().sum()

df_listwise = df_missing.dropna()
df_listwise.shape

file_path = '2_df_train_MAR1_listwise.csv'

# Write the DataFrame to a CSV file
df_listwise.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MAR2MCAR.csv')

df_missing.isnull().sum()

df_listwise = df_missing.dropna()
df_listwise.shape

file_path = '2_df_train_MAR2MCAR_listwise.csv'

# Write the DataFrame to a CSV file
df_listwise.to_csv(file_path, index=False)

… (випущено) …

### 2. Mean imputation

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MCAR5.csv')

df_missing.isnull().sum()

df_mean = df_missing.fillna(df_missing.mean())
df_mean.describe()

file_path = '2_df_train_MCAR5_mean.csv'

# Write the DataFrame to a CSV file
df_mean.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MCAR15.csv')

df_missing.isnull().sum()

df_mean = df_missing.fillna(df_missing.mean())
```

```
df_mean.describe()

file_path = '2_df_train_MCAR15_mean.csv'

# Write the DataFrame to a CSV file
df_mean.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MAR1.csv')

df_missing.isnull().sum()

df_mean = df_missing.fillna(df_missing.mean())
df_mean.describe()

file_path = '2_df_train_MAR1_mean.csv'

# Write the DataFrame to a CSV file
df_mean.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MAR2MCAR.csv')

df_missing.isnull().sum()

df_mean = df_missing.fillna(df_missing.mean())
df_mean.describe()

file_path = '2_df_train_MAR2MCAR_mean.csv'

# Write the DataFrame to a CSV file
df_mean.to_csv(file_path, index=False)

… (випущено) …

### 3. Iterative Imputer (sklearn)

from sklearn.experimental import enable_iterative_imputer  # Enable
IterativeImputer
from sklearn.impute import IterativeImputer

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MCAR5.csv')

df_missing.isnull().sum()

imputer = IterativeImputer(max_iter=10, random_state=0)
imputer.fit(df_missing)
imputed_df = imputer.transform(df_missing)
imputed_df = pd.DataFrame(imputed_df, columns=df_missing.columns)
imputed_df.describe()

file_path = '2_df_train_MCAR5_iterative.csv'
```

```
# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MCAR15.csv')

df_missing.isnull().sum()

imputer = IterativeImputer(max_iter=10, random_state=0)
imputer.fit(df_missing)
imputed_df = imputer.transform(df_missing)
imputed_df = pd.DataFrame(imputed_df, columns=df_missing.columns)
imputed_df.describe()

file_path = '2_df_train_MCAR15_iterative.csv'

# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MAR1.csv')

df_missing.isnull().sum()

imputer = IterativeImputer(max_iter=10, random_state=0)
imputer.fit(df_missing)
imputed_df = imputer.transform(df_missing)
imputed_df = pd.DataFrame(imputed_df, columns=df_missing.columns)
imputed_df.describe()

file_path = '2_df_train_MAR1_iterative.csv'

# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)

df_missing = pd.read_csv('./3-missing-csvs/2_df_train_MAR2MCAR.csv')

df_missing.isnull().sum()

imputer = IterativeImputer(max_iter=10, random_state=0)
imputer.fit(df_missing)
imputed_df = imputer.transform(df_missing)
imputed_df = pd.DataFrame(imputed_df, columns=df_missing.columns)
imputed_df.describe()

file_path = '2_df_train_MAR2MCAR_iterative.csv'

# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)

… (випущено) …

### 4. MICE imputation (using R package mice)
```

```
imputed_df = pd.read_csv('2_df_train_MCAR5_mice.csv')

imputed_df.isnull().sum()

imputed_df = imputed_df.drop(imputed_df.columns[0], axis=1)

imputed_df.describe()

file_path = '2_df_train_MCAR5_mice_fixed.csv'

# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)

imputed_df = pd.read_csv('./3.5-mice-imputed/2_df_train_MCAR15_mice.csv')

imputed_df.isnull().sum()

imputed_df = imputed_df.drop(imputed_df.columns[0], axis=1)

imputed_df.describe()

file_path = '2_df_train_MCAR15_mice_fixed.csv'

# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)

imputed_df = pd.read_csv('./3.5-mice-imputed/2_df_train_MAR1_mice.csv')

imputed_df.isnull().sum()

imputed_df = imputed_df.drop(imputed_df.columns[0], axis=1)
imputed_df.describe()

file_path = '2_df_train_MAR1_mice_fixed.csv'

# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)

imputed_df = pd.read_csv('./3.5-mice-imputed/2_df_train_MAR2MCAR_mice.csv')

imputed_df.isnull().sum()

imputed_df = imputed_df.drop(imputed_df.columns[0], axis=1)
imputed_df.describe()

file_path = '2_df_train_MAR2MCAR_mice_fixed.csv'

# Write the DataFrame to a CSV file
imputed_df.to_csv(file_path, index=False)
```

… (випущено) …

**5. Imputation using R MICE**

```
install.packages("mice")
library(mice)
install.packages("sjmisc")
library(sjmisc)

### Imputation using mice
#### MCAR5
data <- read.csv("3-missing-csvs/2_df_train_MCAR5.csv")
imp_data <- mice(data)
merged_dataset <- merge_imputations(data, imp_data)
data$CreditScore <- merged_dataset$CreditScore
data$Age <- merged_dataset$Age
data$Balance <- merged_dataset$Balance
data$EstimatedSalary <- merged_dataset$EstimatedSalary
write.csv(data, file = '2_df_train_MCAR5_mice.csv', row.names = TRUE)

#### MCAR15
data <- read.csv("3-missing-csvs/2_df_train_MCAR15.csv")
imp_data <- mice(data)
merged_dataset <- merge_imputations(data, imp_data)
data$CreditScore <- merged_dataset$CreditScore
data$Age <- merged_dataset$Age
data$Balance <- merged_dataset$Balance
data$EstimatedSalary <- merged_dataset$EstimatedSalary
write.csv(data, file = '2_df_train_MCAR15_mice.csv', row.names = TRUE)

#### MAR1
data <- read.csv("3-missing-csvs/2_df_train_MAR1.csv")
imp_data <- mice(data)
merged_dataset <- merge_imputations(data, imp_data)
data$CreditScore <- merged_dataset$CreditScore
data$Balance <- merged_dataset$Balance
data$EstimatedSalary <- merged_dataset$EstimatedSalary
write.csv(data, file = '2_df_train_MAR1_mice.csv', row.names = TRUE)

#### MAR2MCAR
data <- read.csv("3-missing-csvs/2_df_train_MAR2MCAR.csv")
imp_data <- mice(data)
merged_dataset <- merge_imputations(data, imp_data)
data$CreditScore <- merged_dataset$CreditScore
data$Age <- merged_dataset$Age
data$Balance <- merged_dataset$Balance
data$EstimatedSalary <- merged_dataset$EstimatedSalary
write.csv(data, file = '2_df_train_MAR2MCAR_mice.csv', row.names = TRUE)
```

… (випущено) …

**6. Imputed dataset modelling example**

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import numpy as np
import pandas as pd
pd.set_option('display.precision', 3)

# Data Visualisation Libraries
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

!pip install seaborn --upgrade
import seaborn as sns
sns.set_style('darkgrid')

# Statistics
from scipy.stats import chi2_contingency
from imblearn.over_sampling import SMOTE

# Machine Learning
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import learning_curve

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
VotingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

from sklearn.metrics import accuracy_score, recall_score, precision_score, auc,
roc_auc_score, roc_curve
from sklearn.metrics import confusion_matrix
import scikitplot as skplt

from helperFunctions import  plot_conf_mx, plot_learning_curve
font_size = 13
plt.rcParams['axes.labelsize'] = font_size
plt.rcParams['axes.titlesize'] = font_size + 2
plt.rcParams['xtick.labelsize'] = font_size - 2
plt.rcParams['ytick.labelsize'] = font_size - 2
plt.rcParams['legend.fontsize'] = font_size - 2

colors = ['seagreen', 'coral']
```

```
colors_cat = ['coral', 'brown', 'navy', 'turquoise', 'crimson', 'teal',
'slategrey', 'dodgerblue', 'darkslateblue', 'orchid']
colors_comp = ['coral', 'brown', 'navy', 'turquoise', 'crimson', 'teal',
'slategrey']

random_state = 42
scoring_metric = 'recall'
comparison_dict, comparison_test_dict = {}, {}

print('Defaults set')

def plot_continuous(feature):
    '''Plot a histogram and boxplot for the churned and retained distributions
for the specified feature.'''
    df_func = train_df.copy()
    df_func['Exited'] = df_func['Exited'].astype('category')

    fig, (ax1, ax2) = plt.subplots(2,
                                   figsize=(7, 5),
                                   sharex=True,
                                   gridspec_kw={'height_ratios': (.7, .3)})

    for df, color, label in zip([df_retained, df_churned], colors, ['Retained',
'Churned']):
        sns.histplot(data=df,
                     x=feature,
                     bins=15,
                     color=color,
                     alpha=0.66,
                     edgecolor='firebrick',
                     label=label,
                     kde=False,
                     ax=ax1)
    ax1.legend()

    sns.boxplot(x=feature, y='Exited', data=df_func, palette=colors, ax=ax2)
    ax2.set_ylabel('')
    ax2.set_yticklabels(['Retained', 'Churned'])

    plt.tight_layout();

def plot_categorical(feature):
    '''For a categorical feature, plot a seaborn.countplot for the total counts
of each category next to a barplot for the churn rate.'''
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

    sns.countplot(x=feature,
                  hue='Exited',
                  data=train_df,
                  palette=colors,
                  ax=ax1)
```

```
    ax1.set_ylabel('Count')
    ax1.legend(labels=['Retained', 'Churned'])

    sns.barplot(x=feature,
                y='Exited',
                data=train_df,
                palette=colors_cat,
                ax=ax2)
    ax2.set_ylabel('Churn rate')

    if (feature == 'HasCrCard' or feature == 'IsActiveMember'):
        ax1.set_xticklabels(['No', 'Yes'])
        ax2.set_xticklabels(['No', 'Yes'])

    plt.tight_layout();

def clf_performance(classifier, classifier_name, classifier_name_abv):
    '''Display the overall performance of a classifier with this template.'''
    print('\n', classifier_name)
    print('------------------------------')
    print('   Best Score ({}): '.format(scoring_metric) +
str(np.round(classifier.best_score_, 3)))
    print('   Best Parameters: ')
    for key, value in classifier.best_params_.items():
        print('      {}: {}'.format(key, value))

    y_pred_pp = cross_val_predict(estimator=classifier.best_estimator_,
                                  X=X_train,
                                  y=y_train,
                                  cv=5,
                                  method='predict_proba')[:, 1]
    y_pred = y_pred_pp.round()

    cm = confusion_matrix(y_train, y_pred, normalize='true')

    fpr, tpr, _ = roc_curve(y_train, y_pred_pp)
    comparison_dict[classifier_name_abv] = [
        accuracy_score(y_train, y_pred),
        precision_score(y_train, y_pred),
        recall_score(y_train, y_pred),
        roc_auc_score(y_train, y_pred_pp), fpr, tpr
    ]

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9, 4))

    plot_conf_mx(cm, ax1)
    plot_learning_curve(classifier.best_estimator_, X_train, y_train, ax2)

    plt.tight_layout();

def plot_feature_imp(classifier, classifier_name, color, ax):
```

```
    '''Plot the importance of features for a classifier as a barplot.'''
    importances = pd.DataFrame({'Feature': X_train.columns,
                                'Importance':
np.round(classifier.best_estimator_.feature_importances_, 3)})

    importances = importances.sort_values('Importance',
ascending=True).set_index('Feature')

    importances.plot.barh(color=color,
                          edgecolor='firebrick',
                          legend=False,
                          ax=ax)
    ax.set_title(classifier_name)
    ax.set_xlabel('Importance');

def test_func(classifier, classifier_name, ax):
    '''Assess the performance on the test set and plot the confusion matrix.'''
    y_pred = classifier.predict(X_test)
    cm = confusion_matrix(y_test, y_pred, normalize='true')

    comparison_test_dict[classifier_name] = [accuracy_score(y_test, y_pred),
                                             precision_score(y_test, y_pred),
                                             recall_score(y_test, y_pred)]

    sns.heatmap(cm,
                annot=True,
                annot_kws={'fontsize': 24},
                cmap='BuPu',
                ax=ax)

    ax.set_title(classifier_name)

    ax.set_xlabel('Predicted Label')
    ax.set_xticks([0.5, 1.5])
    ax.set_xticklabels(['Retained', 'Churned'])

    ax.set_ylabel('True Label')
    ax.set_yticks([0.2, 1.4])
    ax.set_yticklabels(['Retained', 'Churned']);

# 1. Listwise Deletion

df = pd.read_csv('./4-imputed-csvs/2_df_train_MCAR5_listwise.csv')

print(df.shape)
df.head()

df.isnull().sum()

train_df, test_df = train_test_split(df, test_size=0.2,
random_state=random_state)
```

```
train_df.reset_index(drop=True, inplace=True)
test_df.reset_index(drop=True, inplace=True)

print('Train set: {} rows x {} columns'.format(train_df.shape[0],
                                               train_df.shape[1]))
print(' Test set: {} rows x {} columns'.format(test_df.shape[0],
                                               test_df.shape[1]))

### Scaling

scaler = StandardScaler()

scl_columns = ['CreditScore', 'Age', 'Balance',
               'EstimatedSalary'
              ]
train_df[scl_columns] = scaler.fit_transform(train_df[scl_columns])

y_train = train_df['Exited']
X_train = train_df.drop('Exited', 1)

### Sampling

y_train.value_counts()

over = SMOTE(sampling_strategy='auto', random_state=random_state)
X_train, y_train = over.fit_resample(X_train, y_train)

y_train.value_counts()

### Models

lr = LogisticRegression(random_state=random_state)

param_grid = {
    'max_iter': [100],
    'penalty': ['l1', 'l2'],
    'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
    'solver': ['lbfgs', 'liblinear']
}

lr_clf = GridSearchCV(estimator=lr,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_lr_clf = lr_clf.fit(X_train, y_train)
clf_performance(best_lr_clf, 'Logistic Regression', 'LR')
```

```
svc = SVC(probability=True, random_state=random_state)

param_grid = tuned_parameters = [{'kernel': ['rbf'],
                                  'gamma': ['scale', 'auto'],
                                  'C': [.1, 1, 2]},
                                 {'kernel': ['linear'],
                                  'C': [.1, 1, 10]}
                                ]

svc_clf = GridSearchCV(estimator=svc,
                       param_grid=param_grid,
                       scoring=scoring_metric,
                       cv=5,
                       verbose=False,
                       n_jobs=-1)

best_svc_clf = svc_clf.fit(X_train, y_train)
clf_performance(best_svc_clf, 'Support Vector Classifier', 'SVC')

rf = RandomForestClassifier(random_state=random_state)
param_grid = {
    'n_estimators': [100],
    'criterion': ['entropy', 'gini'],
    'bootstrap': [True, False],
    'max_depth': [6],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [2, 3, 5],
    'min_samples_split': [2, 3, 5]
}

rf_clf = GridSearchCV(estimator=rf,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_rf_clf = rf_clf.fit(X_train, y_train)
clf_performance(best_rf_clf, 'Random Forest', 'RF')

lgbmc = LGBMClassifier(random_state=random_state)

param_grid = {
    'max_depth': [5],
    'num_leaves': [5, 10],
    'learning_rate': [0.001, 0.01],
    'n_estimators': [200],
    'feature_fraction': [0.5],
    'min_child_samples': [5, 10],
    'reg_alpha': [0.1, 0.5],
    'reg_lambda': [0.1, 0.5]
```

```
}

lgbmc_clf = GridSearchCV(estimator=lgbmc,
                         param_grid=param_grid,
                         scoring=scoring_metric,
                         cv=5,
                         verbose=False,
                         n_jobs=-1)

best_lgbmc_clf = lgbmc_clf.fit(X_train, y_train)
clf_performance(best_lgbmc_clf, 'LGBMClassifier', 'LGBMC')

### Feature importance

colors_fi = ['steelblue', 'darkgray', 'cadetblue', 'bisque']

fig = plt.subplots(1, 1, figsize=(12, 10))

for i, (name, clf) in enumerate(zip(['RF', 'LGBM'],
                                    [best_rf_clf,
                                     best_lgbmc_clf])):

    ax = plt.subplot(2, 2, i + 1)
    plot_feature_imp(clf, name, colors_fi[i], ax)
    plt.ylabel('')

plt.tight_layout();

### Performance comparison

comparison_matrix = {}
for key, value in comparison_dict.items():
    comparison_matrix[str(key)] = value[0:4]

comparison_df = pd.DataFrame(comparison_matrix,
                             index=['Accuracy', 'Precision', 'Recall', 'AUC']).T
comparison_df.style.highlight_max(color='indianred', axis=0)

comparison_df.plot(kind='bar',
                   figsize=(10, 5),
                   fontsize=12,
                   color=['#5081DE', '#A7AABD', '#D85870', '#424656'])

plt.legend(loc='upper center',
           fontsize=font_size - 6,
           ncol=len(comparison_df.columns),
           bbox_to_anchor=(0.5, 1.12))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
```

```
plt.text(x=-0.5, y=0.73, s='0.70', size=font_size + 2, color='red');

fig, ax = plt.subplots(figsize=(10, 5))

for index, key in enumerate(comparison_dict.keys()):
    auc, fpr, tpr = comparison_dict[key][3], comparison_dict[key][4],
comparison_dict[key][5]
    ax.plot(fpr,
            tpr,
            color=colors_comp[index],
            label='{}: {}'.format(key, np.round(auc, 3)))

ax.plot([0, 1], [0, 1], 'k--', label='Baseline')

ax.set_title('ROC Curve')
ax.set_xlabel('False Positive Rate')
ax.set_xticks([0, 0.25, 0.5, 0.75, 1])
ax.set_ylabel('False Positive Rate')
ax.set_yticks([0, 0.25, 0.5, 0.75, 1])
ax.autoscale(axis='both', tight=True)
ax.legend(fontsize=14);

### Test set

train_df.head()

test_df[scl_columns] = scaler.transform(test_df[scl_columns])  # not
fit_transform, scaler has already been trained

y_test = test_df['Exited']
X_test = test_df.drop('Exited', 1)

print('Preprocessing Complete!')

test_df.shape

# tuned_voting_soft.fit(X_train, y_train)

fig, ax = plt.subplots(4, 1, figsize=(5, 15))

for i, (name, clf) in enumerate(zip(['LR', 'SVC', 'RF',
                                     'LGBM',
                                    ],
                                    [best_lr_clf.best_estimator_,
                                     best_svc_clf.best_estimator_,
                                     best_rf_clf.best_estimator_,
                                     best_lgbmc_clf.best_estimator_,
                                    ])):
    print(clf)
    test_func(clf, name, ax=ax[i])
```

```
plt.tight_layout();

comparison_test_df = pd.DataFrame(comparison_test_dict,
                                  index=['Accuracy', 'Precision', 'Recall']).T
comparison_test_df.style.highlight_max(color='indianred', axis=0)

comparison_test_df.plot(kind='bar',
                        figsize=(10, 5),
                        fontsize=12,
                        color=['#5081DE', '#A7AABD', '#D85870'])

plt.legend(loc='upper center',
           ncol=len(comparison_test_df.columns),
           bbox_to_anchor=(0.5, 1.11))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.72, s='0.70', size=font_size + 2, color='red');

# 2. Mean Imputation

df = pd.read_csv('./4-imputed-csvs/2_df_train_MCAR5_mean.csv')

print('Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df.shape[0], df.shape[1]))
df.head()

df.isnull().sum()

train_df, test_df = train_test_split(df, test_size=0.2,
random_state=random_state)

train_df.reset_index(drop=True, inplace=True)
test_df.reset_index(drop=True, inplace=True)

print('Train set: {} rows x {} columns'.format(train_df.shape[0],
                                               train_df.shape[1]))
print(' Test set: {} rows x {} columns'.format(test_df.shape[0],
                                               test_df.shape[1]))

### Scaling

scaler = StandardScaler()

scl_columns = ['CreditScore', 'Age', 'Balance',
               'EstimatedSalary'
              ]
train_df[scl_columns] = scaler.fit_transform(train_df[scl_columns])

print('Features Scaled!')
```

```
y_train = train_df['Exited']
X_train = train_df.drop('Exited', 1)

print('Sets Created!')

### Sampling

y_train.value_counts()

over = SMOTE(sampling_strategy='auto', random_state=random_state)
X_train, y_train = over.fit_resample(X_train, y_train)

y_train.value_counts()

### Models

lr = LogisticRegression(random_state=random_state)

param_grid = {
    'max_iter': [100],
    'penalty': ['l1', 'l2'],
    'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
    'solver': ['lbfgs', 'liblinear']
}

lr_clf = GridSearchCV(estimator=lr,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_lr_clf = lr_clf.fit(X_train, y_train)
clf_performance(best_lr_clf, 'Logistic Regression', 'LR')

svc = SVC(probability=True, random_state=random_state)

param_grid = tuned_parameters = [{'kernel': ['rbf'],
                                  'gamma': ['scale', 'auto'],
                                  'C': [.1, 1, 2]},
                                 {'kernel': ['linear'],
                                  'C': [.1, 1, 10]}
                                ]

svc_clf = GridSearchCV(estimator=svc,
                       param_grid=param_grid,
                       scoring=scoring_metric,
                       cv=5,
                       verbose=False,
                       n_jobs=-1)
```

```
best_svc_clf = svc_clf.fit(X_train, y_train)
clf_performance(best_svc_clf, 'Support Vector Classifier', 'SVC')

rf = RandomForestClassifier(random_state=random_state)
param_grid = {
    'n_estimators': [100],
    'criterion': ['entropy', 'gini'],
    'bootstrap': [True, False],
    'max_depth': [6],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [2, 3, 5],
    'min_samples_split': [2, 3, 5]
}

rf_clf = GridSearchCV(estimator=rf,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_rf_clf = rf_clf.fit(X_train, y_train)
clf_performance(best_rf_clf, 'Random Forest', 'RF')

lgbmc = LGBMClassifier(random_state=random_state)

param_grid = {
    'max_depth': [5],
    'num_leaves': [5, 10],
    'learning_rate': [0.001, 0.01],
    'n_estimators': [200],
    'feature_fraction': [0.5],
    'min_child_samples': [5, 10],
    'reg_alpha': [0.1, 0.5],
    'reg_lambda': [0.1, 0.5]
}

lgbmc_clf = GridSearchCV(estimator=lgbmc,
                         param_grid=param_grid,
                         scoring=scoring_metric,
                         cv=5,
                         verbose=False,
                         n_jobs=-1)

best_lgbmc_clf = lgbmc_clf.fit(X_train, y_train)
clf_performance(best_lgbmc_clf, 'LGBMClassifier', 'LGBMC')

### Feature importance

colors_fi = ['steelblue', 'darkgray', 'cadetblue', 'bisque']
```

```
fig = plt.subplots(1, 1, figsize=(12, 10))

for i, (name, clf) in enumerate(zip(['RF', 'LGBM'],
                                    [best_rf_clf,
                                     best_lgbmc_clf])):

    ax = plt.subplot(2, 2, i + 1)
    plot_feature_imp(clf, name, colors_fi[i], ax)
    plt.ylabel('')

plt.tight_layout();

### Performance comparison

comparison_matrix = {}
for key, value in comparison_dict.items():
    comparison_matrix[str(key)] = value[0:4]

comparison_df = pd.DataFrame(comparison_matrix,
                             index=['Accuracy', 'Precision', 'Recall', 'AUC']).T
comparison_df.style.highlight_max(color='indianred', axis=0)

comparison_df.plot(kind='bar',
                   figsize=(10, 5),
                   fontsize=12,
                   color=['#5081DE', '#A7AABD', '#D85870', '#424656'])

plt.legend(loc='upper center',
           fontsize=font_size - 6,
           ncol=len(comparison_df.columns),
           bbox_to_anchor=(0.5, 1.12))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.73, s='0.70', size=font_size + 2, color='red');

fig, ax = plt.subplots(figsize=(10, 5))

for index, key in enumerate(comparison_dict.keys()):
    auc, fpr, tpr = comparison_dict[key][3], comparison_dict[key][4],
comparison_dict[key][5]
    ax.plot(fpr,
            tpr,
            color=colors_comp[index],
            label='{}: {}'.format(key, np.round(auc, 3)))

ax.plot([0, 1], [0, 1], 'k--', label='Baseline')

ax.set_title('ROC Curve')
```

```
ax.set_xlabel('False Positive Rate')
ax.set_xticks([0, 0.25, 0.5, 0.75, 1])
ax.set_ylabel('False Positive Rate')
ax.set_yticks([0, 0.25, 0.5, 0.75, 1])
ax.autoscale(axis='both', tight=True)
ax.legend(fontsize=14);

### Test set

train_df.head()

test_df[scl_columns] = scaler.transform(test_df[scl_columns])  # not
fit_transform, scaler has already been trained

y_test = test_df['Exited']
X_test = test_df.drop('Exited', 1)

print('Preprocessing Complete!')

test_df.shape

# tuned_voting_soft.fit(X_train, y_train)

fig, ax = plt.subplots(4, 1, figsize=(5, 15))

for i, (name, clf) in enumerate(zip(['LR', 'SVC', 'RF',
                                     'LGBM',
                                    ],
                                    [best_lr_clf.best_estimator_,
                                     best_svc_clf.best_estimator_,
                                     best_rf_clf.best_estimator_,
                                     best_lgbmc_clf.best_estimator_,
                                    ])):
    print(clf)
    test_func(clf, name, ax=ax[i])

plt.tight_layout();

comparison_test_df = pd.DataFrame(comparison_test_dict,
                                  index=['Accuracy', 'Precision', 'Recall']).T
comparison_test_df.style.highlight_max(color='indianred', axis=0)

comparison_test_df.plot(kind='bar',
                        figsize=(10, 5),
                        fontsize=12,
                        color=['#5081DE', '#A7AABD', '#D85870'])

plt.legend(loc='upper center',
           ncol=len(comparison_test_df.columns),
           bbox_to_anchor=(0.5, 1.11))
plt.xticks(rotation=0)
```

```
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.72, s='0.70', size=font_size + 2, color='red');

# 3. Iterative Imputer

df = pd.read_csv('./4-imputed-csvs/2_df_train_MCAR5_iterative.csv')

print('Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df.shape[0], df.shape[1]))
df.head()

df.isnull().sum()

train_df, test_df = train_test_split(df, test_size=0.2,
random_state=random_state)

train_df.reset_index(drop=True, inplace=True)
test_df.reset_index(drop=True, inplace=True)

print('Train set: {} rows x {} columns'.format(train_df.shape[0],
                                               train_df.shape[1]))
print(' Test set: {} rows x {} columns'.format(test_df.shape[0],
                                               test_df.shape[1]))

### Scaling

scaler = StandardScaler()

scl_columns = ['CreditScore', 'Age', 'Balance',
               'EstimatedSalary'
              ]
train_df[scl_columns] = scaler.fit_transform(train_df[scl_columns])

print('Features Scaled!')

y_train = train_df['Exited']
X_train = train_df.drop('Exited', 1)

print('Sets Created!')

### Sampling

y_train.value_counts()

over = SMOTE(sampling_strategy='auto', random_state=random_state)
X_train, y_train = over.fit_resample(X_train, y_train)

y_train.value_counts()
```

```
### Models

lr = LogisticRegression(random_state=random_state)

param_grid = {
    'max_iter': [100],
    'penalty': ['l1', 'l2'],
    'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
    'solver': ['lbfgs', 'liblinear']
}

lr_clf = GridSearchCV(estimator=lr,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_lr_clf = lr_clf.fit(X_train, y_train)
clf_performance(best_lr_clf, 'Logistic Regression', 'LR')

svc = SVC(probability=True, random_state=random_state)

param_grid = tuned_parameters = [{'kernel': ['rbf'],
                                  'gamma': ['scale', 'auto'],
                                  'C': [.1, 1, 2]},
                                 {'kernel': ['linear'],
                                  'C': [.1, 1, 10]}
                                ]

svc_clf = GridSearchCV(estimator=svc,
                       param_grid=param_grid,
                       scoring=scoring_metric,
                       cv=5,
                       verbose=False,
                       n_jobs=-1)

best_svc_clf = svc_clf.fit(X_train, y_train)
clf_performance(best_svc_clf, 'Support Vector Classifier', 'SVC')

rf = RandomForestClassifier(random_state=random_state)
param_grid = {
    'n_estimators': [100],
    'criterion': ['entropy', 'gini'],
    'bootstrap': [True, False],
    'max_depth': [6],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [2, 3, 5],
    'min_samples_split': [2, 3, 5]
}
```

```
rf_clf = GridSearchCV(estimator=rf,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_rf_clf = rf_clf.fit(X_train, y_train)
clf_performance(best_rf_clf, 'Random Forest', 'RF')

lgbmc = LGBMClassifier(random_state=random_state)

param_grid = {
    'max_depth': [5],
    'num_leaves': [5, 10],
    'learning_rate': [0.001, 0.01],
    'n_estimators': [200],
    'feature_fraction': [0.5],
    'min_child_samples': [5, 10],
    'reg_alpha': [0.1, 0.5],
    'reg_lambda': [0.1, 0.5]
}

lgbmc_clf = GridSearchCV(estimator=lgbmc,
                         param_grid=param_grid,
                         scoring=scoring_metric,
                         cv=5,
                         verbose=False,
                         n_jobs=-1)

best_lgbmc_clf = lgbmc_clf.fit(X_train, y_train)
clf_performance(best_lgbmc_clf, 'LGBMClassifier', 'LGBMC')

### Feature importance

colors_fi = ['steelblue', 'darkgray', 'cadetblue', 'bisque']

fig = plt.subplots(1, 1, figsize=(12, 10))

for i, (name, clf) in enumerate(zip(['RF', 'GB', 'XGB', 'LGBM'],
                                    [best_rf_clf,
#                                       best_gbc_clf,
#                                       best_xgb_clf,
                                     best_lgbmc_clf])):

    ax = plt.subplot(2, 2, i + 1)
    plot_feature_imp(clf, name, colors_fi[i], ax)
    plt.ylabel('')

plt.tight_layout();
```

```
### Performance comparison

comparison_matrix = {}
for key, value in comparison_dict.items():
    comparison_matrix[str(key)] = value[0:4]

comparison_df = pd.DataFrame(comparison_matrix,
                             index=['Accuracy', 'Precision', 'Recall', 'AUC']).T
comparison_df.style.highlight_max(color='indianred', axis=0)

comparison_df.plot(kind='bar',
                   figsize=(10, 5),
                   fontsize=12,
                   color=['#5081DE', '#A7AABD', '#D85870', '#424656'])

plt.legend(loc='upper center',
           fontsize=font_size - 6,
           ncol=len(comparison_df.columns),
           bbox_to_anchor=(0.5, 1.12))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.73, s='0.70', size=font_size + 2, color='red');

fig, ax = plt.subplots(figsize=(10, 5))

for index, key in enumerate(comparison_dict.keys()):
    auc, fpr, tpr = comparison_dict[key][3], comparison_dict[key][4],
comparison_dict[key][5]
    ax.plot(fpr,
            tpr,
            color=colors_comp[index],
            label='{}: {}'.format(key, np.round(auc, 3)))

ax.plot([0, 1], [0, 1], 'k--', label='Baseline')

ax.set_title('ROC Curve')
ax.set_xlabel('False Positive Rate')
ax.set_xticks([0, 0.25, 0.5, 0.75, 1])
ax.set_ylabel('False Positive Rate')
ax.set_yticks([0, 0.25, 0.5, 0.75, 1])
ax.autoscale(axis='both', tight=True)
ax.legend(fontsize=14);

### Test set

train_df.head()

test_df[scl_columns] = scaler.transform(test_df[scl_columns])  # not
fit_transform, scaler has already been trained
```

```
y_test = test_df['Exited']
X_test = test_df.drop('Exited', 1)

print('Preprocessing Complete!')

test_df.shape

fig, ax = plt.subplots(4, 1, figsize=(5, 15))

for i, (name, clf) in enumerate(zip(['LR', 'SVC', 'RF',
                                     'LGBM',
                                    ],
                                    [best_lr_clf.best_estimator_,
                                     best_svc_clf.best_estimator_,
                                     best_rf_clf.best_estimator_,
                                     best_lgbmc_clf.best_estimator_,
                                    ])):
    print(clf)
    test_func(clf, name, ax=ax[i])

plt.tight_layout();

comparison_test_df = pd.DataFrame(comparison_test_dict,
                                  index=['Accuracy', 'Precision', 'Recall']).T
comparison_test_df.style.highlight_max(color='indianred', axis=0)

comparison_test_df.plot(kind='bar',
                        figsize=(10, 5),
                        fontsize=12,
                        color=['#5081DE', '#A7AABD', '#D85870'])

plt.legend(loc='upper center',
           ncol=len(comparison_test_df.columns),
           bbox_to_anchor=(0.5, 1.11))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.72, s='0.70', size=font_size + 2, color='red');

# 4. R MICE Imputation

df = pd.read_csv('./4-imputed-csvs/2_df_train_MCAR5_mice_fixed.csv')

print('Dataset Imported Successfully!\n')
print('It contains {} rows and {} columns.'.format(df.shape[0], df.shape[1]))
df.head()

df.isnull().sum()
```

```
train_df, test_df = train_test_split(df, test_size=0.2,
random_state=random_state)

train_df.reset_index(drop=True, inplace=True)
test_df.reset_index(drop=True, inplace=True)

print('Train set: {} rows x {} columns'.format(train_df.shape[0],
                                               train_df.shape[1]))
print(' Test set: {} rows x {} columns'.format(test_df.shape[0],
                                               test_df.shape[1]))

### Scaling

scaler = StandardScaler()

scl_columns = ['CreditScore', 'Age', 'Balance',
               'EstimatedSalary'
              ]
train_df[scl_columns] = scaler.fit_transform(train_df[scl_columns])

print('Features Scaled!')

y_train = train_df['Exited']
X_train = train_df.drop('Exited', 1)

print('Sets Created!')

### Sampling
y_train.value_counts()

over = SMOTE(sampling_strategy='auto', random_state=random_state)
X_train, y_train = over.fit_resample(X_train, y_train)

y_train.value_counts()

### Models

lr = LogisticRegression(random_state=random_state)

param_grid = {
    'max_iter': [100],
    'penalty': ['l1', 'l2'],
    'C': [0.0001, 0.001, 0.01, 0.1, 1, 10],
    'solver': ['lbfgs', 'liblinear']
}

lr_clf = GridSearchCV(estimator=lr,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
```

```
                      n_jobs=-1)

best_lr_clf = lr_clf.fit(X_train, y_train)
clf_performance(best_lr_clf, 'Logistic Regression', 'LR')

svc = SVC(probability=True, random_state=random_state)

param_grid = tuned_parameters = [{'kernel': ['rbf'],
                                  'gamma': ['scale', 'auto'],
                                  'C': [.1, 1, 2]},
                                 {'kernel': ['linear'],
                                  'C': [.1, 1, 10]}
                                ]

svc_clf = GridSearchCV(estimator=svc,
                       param_grid=param_grid,
                       scoring=scoring_metric,
                       cv=5,
                       verbose=False,
                       n_jobs=-1)

best_svc_clf = svc_clf.fit(X_train, y_train)
clf_performance(best_svc_clf, 'Support Vector Classifier', 'SVC')

rf = RandomForestClassifier(random_state=random_state)
param_grid = {
    'n_estimators': [100],
    'criterion': ['entropy', 'gini'],
    'bootstrap': [True, False],
    'max_depth': [6],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [2, 3, 5],
    'min_samples_split': [2, 3, 5]
}

rf_clf = GridSearchCV(estimator=rf,
                      param_grid=param_grid,
                      scoring=scoring_metric,
                      cv=5,
                      verbose=False,
                      n_jobs=-1)

best_rf_clf = rf_clf.fit(X_train, y_train)
clf_performance(best_rf_clf, 'Random Forest', 'RF')

lgbmc = LGBMClassifier(random_state=random_state)

param_grid = {
    'max_depth': [5],
    'num_leaves': [5, 10],
    'learning_rate': [0.001, 0.01],
```

```
    'n_estimators': [200],
    'feature_fraction': [0.5],
    'min_child_samples': [5, 10],
    'reg_alpha': [0.1, 0.5],
    'reg_lambda': [0.1, 0.5]
}

lgbmc_clf = GridSearchCV(estimator=lgbmc,
                         param_grid=param_grid,
                         scoring=scoring_metric,
                         cv=5,
                         verbose=False,
                         n_jobs=-1)

best_lgbmc_clf = lgbmc_clf.fit(X_train, y_train)
clf_performance(best_lgbmc_clf, 'LGBMClassifier', 'LGBMC')

### Feature importance

colors_fi = ['steelblue', 'darkgray', 'cadetblue', 'bisque']

fig = plt.subplots(1, 1, figsize=(12, 10))

for i, (name, clf) in enumerate(zip(['RF', 'GB', 'XGB', 'LGBM'],
                                    [best_rf_clf,
#                                       best_gbc_clf,
#                                       best_xgb_clf,
                                     best_lgbmc_clf])):

    ax = plt.subplot(2, 2, i + 1)
    plot_feature_imp(clf, name, colors_fi[i], ax)
    plt.ylabel('')

plt.tight_layout();

### Performance comparison

comparison_matrix = {}
for key, value in comparison_dict.items():
    comparison_matrix[str(key)] = value[0:4]

comparison_df = pd.DataFrame(comparison_matrix,
                             index=['Accuracy', 'Precision', 'Recall', 'AUC']).T
comparison_df.style.highlight_max(color='indianred', axis=0)

comparison_df.plot(kind='bar',
                   figsize=(10, 5),
                   fontsize=12,
                   color=['#5081DE', '#A7AABD', '#D85870', '#424656'])

plt.legend(loc='upper center',
```

```
           fontsize=font_size - 6,
           ncol=len(comparison_df.columns),
           bbox_to_anchor=(0.5, 1.12))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.73, s='0.70', size=font_size + 2, color='red');

fig, ax = plt.subplots(figsize=(10, 5))

for index, key in enumerate(comparison_dict.keys()):
    auc, fpr, tpr = comparison_dict[key][3], comparison_dict[key][4],
comparison_dict[key][5]
    ax.plot(fpr,
            tpr,
            color=colors_comp[index],
            label='{}: {}'.format(key, np.round(auc, 3)))

ax.plot([0, 1], [0, 1], 'k--', label='Baseline')

ax.set_title('ROC Curve')
ax.set_xlabel('False Positive Rate')
ax.set_xticks([0, 0.25, 0.5, 0.75, 1])
ax.set_ylabel('False Positive Rate')
ax.set_yticks([0, 0.25, 0.5, 0.75, 1])
ax.autoscale(axis='both', tight=True)
ax.legend(fontsize=14);

### Test set

train_df.head()

test_df[scl_columns] = scaler.transform(test_df[scl_columns])  # not
fit_transform, scaler has already been trained

y_test = test_df['Exited']
X_test = test_df.drop('Exited', 1)

print('Preprocessing Complete!')

test_df.shape

fig, ax = plt.subplots(4, 1, figsize=(5, 15))

for i, (name, clf) in enumerate(zip(['LR', 'SVC', 'RF',
                                     'LGBM',
                                    ],
                                    [best_lr_clf.best_estimator_,
                                     best_svc_clf.best_estimator_,
                                     best_rf_clf.best_estimator_,
```

```
                                    best_lgbmc_clf.best_estimator_,
                                   ])):
    print(clf)
    test_func(clf, name, ax=ax[i])

plt.tight_layout();

comparison_test_df = pd.DataFrame(comparison_test_dict,
                                  index=['Accuracy', 'Precision', 'Recall']).T
comparison_test_df.style.highlight_max(color='indianred', axis=0)

comparison_test_df.plot(kind='bar',
                        figsize=(10, 5),
                        fontsize=12,
                        color=['#5081DE', '#A7AABD', '#D85870'])

plt.legend(loc='upper center',
           ncol=len(comparison_test_df.columns),
           bbox_to_anchor=(0.5, 1.11))
plt.xticks(rotation=0)
plt.yticks([0, 0.4, 0.8])

plt.axhline(y=0.70, color='red', linestyle='--')
plt.text(x=-0.5, y=0.72, s='0.70', size=font_size + 2, color='red');
```