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

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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСИФІКАЦІЇ СЕРВІСІВ E-COMMERCE КОМПАНІЇ НА ОСНОВІ АНСАМБЛІВ МОДЕЛЕЙ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.22150102

Виконав студент 6-го курсу, групи 601
О.О. Жебко
(підпис, ініціали та прізвище)
«___» _____202_р.

Керівник: **д-р. техн. наук, професор**
(наук. ступінь, вчене звання)
О.П. Гожий
(підпис, ініціали та прізвище)
«___» _____202_р.

Миколаїв – 2023

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

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

ЗАВДАННЯ

на магістерську кваліфікаційну роботу

Жебку Олександру Олеговичу
(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи «Інтелектуальна система класифікації сервісів E-Commerce компанії на основі ансамблів моделей».
Керівник роботи Гожий Олександр Петрович, д-р техн. наук, професор.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» листопада 2022 р. № 199
2. Строк подання студентом роботи 15 лютого 2023 р.
3. Вхідні (початкові) дані до роботи: Проаналізуємо набір даних, наданий міжнародною компанією, яка хоче отримати дані сервісу доставки зі своєї бази даних клієнтів. Набір даних, використаний для побудови моделі, містить 10999 спостережень та 12 змінних. Дані містять таку інформацію як: ідентифікаційний номер клієнтів, складський блок, спосіб доставки, дзвінки служби підтримки клієнтів, рейтинг клієнтів, вартість продукту, попередні покупки, важливість продукту, стать покупця, знижка, вага в грамах, термін доставки.
Очікуваний результат: інтелектуальна система класифікації сервісів E-Commerce компанії.
4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):
   - класифікація інтелектуальних систем та їх можливостей;

– дослідження використання задач прогнозування у керуванні компанією;

– вивчення та дослідження методів ансамблювання моделей та засобів підготовки даних та метрик якості;

– виконання попередньої обробки даних та їх аналіз;

– вибір та реалізація базових алгоритмів для слабких моделей;

– розробка першого шару ансамблювання, що складається з набору базових моделей;

– оцінка метрик якості моделей першого шару та їх використання для навчання моделі другого шару ансамблю;

– розробка другого шару ансамблювання та оцінка ефективності метрик якості моделей другого шару;

– порівняльний аналіз отриманих результатів.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: аналіз вимог до умов праці, заходи з техніки безпеки та охорона праці під час роботи у випадку настання надзвичайних ситуацій.

7. Консультанти:

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

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

_____
(підпис)

Завдання прийнято до виконання ___Жебко О. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «__07__» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

Виконання магістерської кваліфікаційної роботи

Тема: Інтелектуальна система класифікації сервісів E-Commerce компанії на основі ансамблів моделей

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР	01.09.2022	20.10.2022	Виконано
2	Отримання завдання на виконання МКР	21.10.2022	08.11.2022	Виконано
3	Складання календарного плану на період виконання МКР	08.11.2022	13.11.2022	Виконано
4	Огляд літератури за темою дослідження	14.11.2022	27.11.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	28.11.2022	18.12.2022	Виконано
6	Аналіз предметної області та розробка технічного завдання. Моделювання результатів.	19.12.2022	12.01.2023	Виконано
7	Визначення базових методів прогнозування, методів ансамблювання моделей. Виконання аналізу даних та попередньої обробки даних. Реалізація базових класифікаторів та ансамблю моделі дворівневої класифікації. Оцінка ефективності розроблених моделей класифікації.	13.01.2023	25.01.2023	Виконано
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2023	02.02.2023	Виконано
9	Попередній захист МКР на засіданні комісії кафедри	03.02.2023	03.02.2023	Виконано
10	Корегування роботи за результатами попереднього захисту	04.02.2023	06.02.2023	Виконано
11	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	07.02.2023	09.02.2023	Виконано
12	Подання МКР рецензенту	09.02.2023	10.02.2023	Виконано
13	Рецензування МКР	11.02.2023	12.02.2023	Виконано
14	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2023	16.02.2023	Виконано
15	Захист МКР перед екзаменаційною комісією (ЕК)	22.02.2023	23.02.2023	

Розробив студент Жебко О.О. _____

(прізвище та ініціали) (*підпис*)

Керівник роботи д-р. техн. наук, професор Гожий О.П. _____

(*наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«12» листопада 2022 р.

АНОТАЦІЯ

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

Жебка Олександра Олеговича

на тему: **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСИФІКАЦІЇ СЕРВІСІВ E-COMMERCE КОМПАНІЇ НА ОСНОВІ АНСАМБЛІВ МОДЕЛЕЙ”**

Актуальність даного дослідження полягає у необхідності підвищення ефективності сервісів доставки із врахуванням транспортних маршрутів, часових обмежень та характеристик товару, розробці програмного забезпечення з використанням сучасних мета-евристичних методів для вирішення відповідної задачі. Це дозволить зменшити витрати компанії та підвищити якість обслуговування клієнтів.

Об’єктом дослідження є процес аналізу даних про сервіси доставки товарів міжнародної компанії із виявленням закономірностей та взаємозалежностей для подальшого прогнозування.

Предметом дослідження є математичні моделі інтелектуального аналізу даних та їх ансамблі для вирішення задач класифікації на основі статистичних даних.

Метою дослідження є покращення сервісів E-Commerce компанії на основі ансамблів моделей. Необхідно виявити чинники, пов'язані з ризиком невчасної доставки товару клієнту.

В ході виконання магістерської кваліфікаційної роботи було використано методи інтелектуального аналізу даних та інтелектуального прийняття рішень на основі даних навчальної вибірки. Також, розглянуто та проаналізовано одні з найбільш впроваджених з тих, що існують на даний момент, сучасних методів прогнозування, що базуються на деревах рішень, наївному баєсівському класифікаторі, лінійному дискримінантному аналізі, квадратичному дискримінантному аналізі, логістичній регресії, методі опорних векторів, методі

найближчого сусіда, штучних нейронних мережах та моделі випадкового лісу. Для покращення результатів прогнозування використано ансамблеві методи – кілька базових моделей навчались для вирішення однієї і тієї ж проблеми та було об'єднано для отримання кращих результатів. Проведено дослідження базових методів прогнозування та ансамблевих моделей на основі стекінгу та беггінгу, а також метрик оцінки ефективності використання базових класифікаторів та моделей першого та другого рівня, визначено наступні параметри для усіх наведених методів у роботі: точність прогнозування та коефіцієнт помилок, Каппа-статистика, чутливість та специфічність, точність та повнота, F-міра та площею під ROC-кривою.

Дана робота складається з шести розділів. В першому розділі здійснено огляд та класифікацію інтелектуальних систем та їх можливостей для вирішення задач прогнозування, а також, використання задач класифікації у керуванні компанією. У другому розділі описано ключові характеристики якості моделей, вибір метрики, підбір базових (слабких) моделей, підбір параметрів для базових моделей та ансамблевих методів. В третьому розділі описано процес збору даних, їх аналіз та інтерпретація, виконано попередню обробку даних та розділено набір на тренувальну та тестову вибірки та згенеровано вхідні змінні на основі поведінкових даних клієнтів. У четвертому розділі наведено результати застосування простих класифікаторів та ансамблю моделі дворівневої класифікації та виконано оцінку ефективності розроблених моделей класифікації. П’ятий розділ присвячено розробці методичних матеріалів. У шостому розділі наведено основні положення з охорони праці та цивільного захисту під час надзвичайних ситуацій. Загальний обсяг роботи – 175 сторінок. Магістерська кваліфікаційна робота містить один додаток, 22 формули, 59 рисунків, 5 таблиць, посилання на 66 літературних джерел.

Ключові слова: інтелектуальна система, ансамбль моделей, задача прогнозування, базові алгоритми, машинне навчання.

ABSTRACT

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

Zhebko Oleksandr

“INTELLIGENT SYSTEM FOR CLASSIFYING A COMPANY'S E-COMMERCE SERVICES BASED ON ENSEMBLE MODELS”

The relevance of this study lies in the need to improve the efficiency of delivery services, taking into account transport routes, time constraints and product characteristics, and to develop software using modern meta-heuristic methods to solve the corresponding problem. This will reduce the company's benefits and improve the quality of customer service.

The object of the study is the process of analyzing data on the delivery services of an international company with the identification of patterns and interdependencies for further forecasting.

The subject of the study is mathematical models of data mining and their ensembles for solving classification problems based on statistical data.

The aim of the thesis is to improve the company's E-Commerce services based on ensemble models. It is necessary to identify the factors associated with the risk of untimely delivery of goods to the client.

In the course of the master's thesis, the methods of data mining and intelligent decision-making based on the training sample data were used. Also, some of the most widely used modern forecasting methods based on decision trees, naive Bayes classifiers, linear discriminant analysis, quadratic discriminant analysis, logistic regression, support vector machine, k-nearest neighbors algorithm, artificial neural networks, and random forest model were considered and analyzed. To improve the forecasting results, ensemble methods were used - several basic models were trained to solve the same problem and combined to obtain better results. The study of basic forecasting methods and ensemble

models based on stacking and bagging, as well as metrics for evaluating the effectiveness of using basic classifiers and models of the first and second level, was conducted, and the following parameters were determined for all the methods in the paper: prediction accuracy and error rate, Kappa statistic, sensitivity and specificity, precision and recall, F-measure and area under the ROC curve.

This paper consists of six chapters. The first section provides an overview and classification of intelligent systems and their capabilities for solving forecasting tasks, as well as the use of classification tasks in company management. The second section describes the key characteristics of model quality, metric selection, selection of basic (weak) models, selection of parameters for basic models and ensemble methods.

This thesis consists of six chapters. The first section provides an overview and classification of intelligent systems and their capabilities for solving forecasting problems, as well as the use of classification tasks in company management. The second section describes the key characteristics of model quality, metric selection, selection of basic (weak) models, selection of parameters for basic models and ensemble methods. The third section describes the process of data collection, analysis and interpretation, performs data preprocessing and divides the set into training and test samples, and generates input variables based on customer behavioral data. The fourth section presents the results of applying simple classifiers and the ensemble of the two-level classification model and evaluates the effectiveness of the developed classification models. Section 5 presents the main provisions on labor protection and civil defense during emergencies. The sixth section is devoted to the development of methodological materials. The master's qualification work contains 175 pages, one appendix, 22 formulas, 59 figures, 5 tables and 66 references.

Keywords: intelligent system, ensemble model, forecasting task, basic algorithms, machine learning.

Зміст

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСИФІКАЦІЇ СЕРВІСІВ E-COMMERCE КОМПАНІЇ НА ОСНОВІ АНСАМБЛІВ МОДЕЛЕЙ»

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.22150102

Виконав студент 6-го курсу, групи 601
_____ О.О. Жебко
(підпис, ініціали та прізвище)
«___» _____ 2023р.

Керівник: д-р. техн. наук, професор
(наук. ступінь, вчене звання)
_____О.П. Гожий_
(підпис, ініціали та прізвище)
«___» _____ 2023 р.

Миколаїв – 2023

ВСТУП

З розвитком інформаційних технологій поступово збільшується сфера їх вжитку. Сучасні інтелектуальні системи роботи з даними здатні безперервно функціонувати в режимі реального часу, що дає змогу впроваджувати та використовувати різноманітні методи прогнозування. Особливо чутливим до якості прогнозу є застосування цих алгоритмів у сфері менеджменту, що використовує різні методи передбачення для покращення якості керування компанією з метою підвищення якості послуг, що надаються.

Зазвичай для такого аналізу використовується приватна інформація про клієнтів. Перевагою даної роботи є те, що в якості вхідних даних використовується дані сервісів E-Commerce міжнародної компанії з доставки зі своєї бази даних клієнтів, що не є приватними. Однак, цей тип даних має недолік – їх використання в класичних моделях є недостатньо репрезентативним для застосування, тому в цій роботі використано методи ансамблювання слабких моделей з метою одержання сильної на їх основі.

Актуальність даного дослідження полягає у необхідності підвищення ефективності сервісів доставки із врахуванням транспортних маршрутів, часових обмежень та характеристик товару, розробці програмного забезпечення з використанням сучасних мета-евристичних методів для вирішення відповідної задачі. Це дозволить зменшити витрати компанії та підвищити якість обслуговування клієнтів.

Об’єкт дослідження – процес аналізу даних про сервіси доставки товарів міжнародної компанії із виявленням закономірностей та взаємозалежностей для подальшого прогнозування.

Предмет дослідження – є математичні моделі інтелектуального аналізу даних та їх ансамблі для вирішення задач класифікації на основі статистичних даних.

Мета дослідження – покращення сервісів E-Commerce компанії на основі ансамблів моделей. Потрібно виявити чинники, пов'язані з ризиком невчасної

доставки товару клієнту. Для цього необхідно отримати дані про велику кількість попередніх замовлень клієнтами та інформацію про одержувачів товарів.

Методи дослідження – базові моделі прогнозування: дерева рішень (Decision Tree), Наївний Баєсів класифікатор (NB), лінійний дискримінантний аналіз (LDA), квадратичний дискримінантний аналіз (QDA), логістична регресія (LR), метод опорних векторів (SVM), метод найближчого сусіда (KNN), штучні нейронні мережі (ANN) та модель випадкового лісу (RF), багаторівневий ансамбль моделей, Bagging, Stacking.

Виходячи з мети, перед даним дослідженням поставлено **наступні завдання**:

1) вивчити та дослідити методи ансамблювання моделей, засоби підготовки даних та метрик якості;

2) обрати алгоритм та побудувати базові моделі;

3) розробити перший шар, що складається з набору базових моделей;

4) обробити результати моделей першого шару як входи моделі другого шару(ансамбль);

5) проаналізувати результати та створити ансамбль моделей;

6) проаналізувати ефективність ансамблевих моделей.

Апробація результатів дослідження. Основні положення дипломної роботи, зокрема інтелектуальна система класифікації сервісів E-Commerce компанії на основі ансамблів моделей, викладено в матеріалах XXV Всеукраїнської науково-практичної конференції «Могилянські читання-2022. Досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» та Всеукраїнській науково-практичній конференції молодих вчених, аспірантів і студентів «Інформаційні технології та інженерія».

Структура роботи. Магістерська кваліфікаційна робота складається зі вступу, 6 розділів, висновків та списку використаних джерел, що нараховує 66 позицій, 59 рисунків, 22 формул, 5 таблиць та одного додатку. Загальний обсяг 175 сторінок.

1 ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСИФІКАЦІЇ ДЛЯ ОЦІНКИ ЯКОСТІ СЕРВІСІВ E-COMMERCE КОМПАНІЇ

В даний час розвиток науки і техніки досяг такого рівня, коли вже стає реальним створення штучного інтелекту або моделювання (імітація) можливостей і здібностей людини, а вирішення зазначених основних завдань за допомогою програмних та апаратних засобів. Тому вивченню інтелектуальних систем має передувати розгляд основних властивостей та особливостей природного інтелекту для того, щоб зрозуміти та використовувати властивості біологічних систем для вирішення технічних проблем. Кібернетичне вивчення живого допомагає розкрити як загальні закони функціонування складних систем, так і властивості окремих органів та організму загалом, з точки зору інформаційних процесів і процесів управління.

1.1 Класифікація інтелектуальних систем та їх можливостей

Нове покоління систем – інтелектуальні системи (ІС) – породили інші принципи організації компонентів систем, з'явилися інші поняття, терміни, блоки, які раніше не зустрічалися в розробках і, отже, у науковій літературі. Інтелектуальні системи здатні синтезувати ціль, приймати рішення до дії, забезпечувати дію для досягнення мети, прогнозувати значення параметрів результату дії та зіставляти їх із реальними, утворюючи зворотний зв'язок, чи змінювати поставлені цілі [1, с. 12].

ІС можуть вирішувати інтелектуальні завдання, розпізнавати ситуації (образи), навчатися поняттям і навичкам, формувати модель обстановки (розв'язуваної задачі), планувати поведінку (приймати рішення), робити прогнози і здійснювати їх обробку. ІС на вирішення різних завдань залежать, передусім від продуктивності сучасних ЕОМ [1, с. 13].

Характерною рисою вже діючих систем, орієнтованих переважно на обробку знань, є високий рівень розвитку їх програмного забезпечення. З його допомогою

вирішуються завдання обробки символьної інформації, перебору рішень обчислювальних та логічних завдань та побудови логічного висновку, рішення з використанням заданих систем правил, роботи з БД, високошвидкісної обробки зображень, мовлення та інші. В даний час при розробці інтелектуальних систем все частіше використовуються спеціалізовані апаратні засоби, що реалізують у тому чи іншою мірою їх основні функції [2, с. 25].

Вивчення ІС дозволяє зробити спробу сформулювати загальні принципи, які, будучи недостатніми, відображають необхідні моменти в їх організації та функціонуванні [2, с. 47].

- Принцип системності. ІС можуть бути лише складними системами, функції всіх їх елементів повинні бути узгоджені з призначенням системи та їх місцем у них, а також між собою. Саме взаємна узгодженість та взаємозалежність елементів системи забезпечує цілісність та функціональну повноту найбільш досконалих ІС [2, с. 48].

- Принцип ієрархічності. Складна ієрархічна багаторівнева структура є основою одночасного перебігу безлічі процесів. Рівень неординарності підсумкового процесу залежить від характеру сукупності складових процесів. Складна сукупність процесів принципово характеризується складною структурою. Таким чином, певною мірою рівень складності системи та її структури визначає і потенційний рівень її інтелекту [2, с. 48].

- Принцип багатоканальності. Отримання узгоджених з обставинами та середовищем рішень різних задач ґрунтується на інформації, що одержується ззовні по багатьох каналах та працює на різних фізичних принципах, що дозволяє мати різнорідну характеристику спеціальних властивостей об'єктів середовища. Комплексування інформаційних даних дозволяє мати більш об'єктивну і більш повну картину про процеси, що відбуваються. Різнорідна інформація, одержувана різними каналами, обробляється приблизно за однаковий мінімально можливий час [2, с. 49].

Наочність цього принципу характеризує наступний факт. Людина здатна вирішувати різноманітні розпізнавальні задачі за частки секунди, а зорова система людини безсумнівно працює як паралельний пристрій. Паралельна обробка як зорової інформації, так й іншої, що надходить у мозок людини з інших органів чуття, дозволяє реалізувати інваріантне впізнання об'єктів [2, с. 49].

- Принцип адаптивності. Принцип адаптивності передбачає наявність у інтелектуальних систем потенційних можливостей поліпшення роботи: в умовах апріорної та поточної невизначеності на основі навчання на досвіді [2, с. 49].

Особлива роль при цьому належить елементам системи, що реалізують пам'ять. Адаптація може відбуватися шляхом самонавчання чи самоорганізації. Адаптивні здібності можуть визначатися обсягом інформації (пам'яттю) системи та потрібними витратами часу на її обробку [2, с. 55].

- Принцип взаємності функціональних та структурних властивостей. Природно, що призначення системи та її функції безпосередньо впливають на структуру системи. Однак і структура системи має сприяти найбільш повній реалізації функцій [2, с. 55].

- Принцип еквіфінальності. Цей принцип передбачає наявність у системи безлічі взаємоузгоджених послідовностей реакцій на певні зовнішні впливи, що призводять до того самого практично корисного результату [2, с. 56].

- Принцип динамічного самопрограмування. Найбільш чудова особливість нервового управління, найбільш яскраво виражена у цілеспрямованому творчому розумі людини, полягає у здатності на підставі різноманітного аналізу ситуацій миттєво створювати найскладніші і водночас оптимальні програми діяльності, які безперервно перебудовуються та коригуються з урахуванням минулих подій, поточної дійсності та прогнозування майбутнього. Вже створення елементарного умовного рефлексу є вироблення нової програми поведінки. Ускладнення умовних рефлексів означає дедалі вищу самоорганізацію поведінкових програм. У кібернетичному сенсі основна функція вищої нервової діяльності полягає у динамічній поведінці самопрограмування [2, с. 56].

Функціональні можливості інтелектуальних систем. Природні системи розрізняються за своєю складністю та рівнем організації. Поняття про організацію системи передбачає певне узгодження станів та діяльності її підсистем та складових елементів. Це узгодження досягається передачею сигналів (повідомлень) по внутрішньосистемним зв'язкам, а для підтримки високого рівня організованості необхідне постійне спілкування з навколишнім світом. Ще більш необхідна передача повідомлень з внутрішньосистемних та міжсистемних зв'язків для формування та видачі командних сигналів під час здійснення актів управління [3].

Основною властивістю природних ІС є їхня здатність до адаптації при зміні умов функціонування. Здатність до адаптації шляхом самоорганізації ґрунтується як на множинності елементів системи та розгалуженості зв'язків між ними, що сприяють виникненню цілісності, так і на наявності гнучкої взаємодії між елементами на кшталт зворотних зв'язків. Істотною ознакою самоорганізації є відокремлення інтелектуальних систем від довкілля [1, с. 98].

Функціональною особливістю відокремленої ІС є активна взаємодія її із середовищем. Особливості її структурної організації визначають напрям та обсяг процесів взаємодії системи із середовищем. Наявність надзвичайно різноманітних зворотних зв'язків на всіх рівнях впливає на інтенсивність процесів взаємодії. Негативні зворотні зв'язки забезпечують стабільність функцій системи, сталість її параметрів, стійкість до зовнішніх впливів. Позитивні зворотні зв'язки відіграють роль підсилювачів процесів і мають особливе значення для розвитку, накопичення змін. Наявність негативних та позитивних зворотних зв'язків призводить до можливості розвитку за деяким законом (програмою) з використанням зовнішніх ресурсів [1, с. 101].

Складна динамічна (стійко нерівноважна) організація цілеспрямованої функціонуючої системи вимагає безперервного управління без якого система неспроможна існувати. Особливість цього управління полягає в тому, що воно

спричиняє ряд процесів у самій системі і насамперед процесів внутрішнього саморегулювання за законами організації системи [4].

Основними функціями системи, що самоорганізується, є функції інформаційного забезпечення (ФІЗ), матеріального та енергетичного забезпечення (ФМЕЗ), переміщення (ФП) та адаптації (ФА). З погляду реалізації ШІ найбільший інтерес представляє ФІЗ, яка є всеосяжною. Інформація необхідна для контролю внутрішнього стану системи, розпізнавання ситуацій, вирішення задач забезпечення функціонування, виявлення закономірностей і навчання. Для подальшого використання одержувана інформація повинна розділятися та відкладатися у відповідні системи пам'яті (оперативні та довготривалі) [4].

Функцію інформаційного забезпечення реалізують органи контролю довкілля, навігації та аналізу об'єктів. Обробка сигналів цих органів інформації здійснюється особливим керуючим вузлом (пристроєм), у якому проводиться аналіз отриманих даних, їх обробка та узагальнення, оцінка ситуації та прийняття рішення. Одночасно ведеться збагачення пам'яті, накопичення досвіду, навчання та відпрацювання логічних методів обробки інформації [5].

Методи набуття знань. Динамічні властивості інтелектуальних систем можуть бути описані у просторі станів. Інтелектуальні оператори, що реалізують сприйняття, уявлення, формування поняття, судження та умовиводи у процесі пізнання є формальним засобом обробки відомостей та знань, а також прийняття рішення. Всі ці аспекти мають бути покладені в основу побудови ДЕС, що функціонують у реальному часі та реальному світі [5].

Динамічна експертна система є деяке комплексне утворення, здатне оцінювати стан системи та середовища, зіставляти параметри бажаних та реальних результатів дії, приймати рішення та виконувати управління, що сприяє досягненню мети. Для цього ДЕС повинна мати запас знань і мати у своєму розпорядженні методи вирішення завдань [6].

Види інтелектуальних систем. Експертна система (ЕС, expert system) - комп'ютерна програма, здатна частково замінити фахівця-експерта у вирішенні

проблемної ситуації. Експертні системи почали розроблятися дослідниками штучного інтелекту у 1970-х роках, а у 1980-х отримали комерційне підкріплення.

В інформатиці експертні системи розглядаються спільно з базами знань як моделі поведінки експертів у певній галузі знань з використанням процедур логічного висновку та прийняття рішень, а бази знань – як сукупність фактів та правил логічного висновку у вибраній предметній галузі діяльності [2, с. 132].

Подібні дії виконує програма-майстер (wizard). Майстри застосовуються як у системних програмах, так і в прикладних для інтерактивного спілкування з користувачем (наприклад, при встановленні ПЗ). Головна відмінність майстрів від ЕС – відсутність бази знань; всі дії жорстко запрограмовані. Це просто набір форм для заповнення користувачем [2, с. 135].

Інші подібні програми – пошукові або довідкові (енциклопедичні) системи. За запитом користувача вони надають найбільш відповідні (релевантні) розділи бази статей (уявлення про об'єкти знань, їх віртуальну модель) [2, с. 135].

Гібридні інтелектуальні системи (ГІС). Під ГІС прийнято розуміти систему, в якій для вирішення задачі використовується більше одного методу імітації інтелектуальної діяльності. Таким чином, гібридна ІС – це сукупність аналітичних моделей, експертних систем, штучних нейронних мереж, нечітких систем генетичних алгоритмів, імітаційних статистичних моделей [1, с. 122].

Інтелектуально – інформаційні системи. Інтелектуальна інформаційна система (ІІС, англ. intelligent system) – різновид інтелектуальної системи, один з видів інформаційних систем, іноді ІІС називають системою, заснованою на знаннях. ІІС є комплексом програмних, лінгвістичних та логіко-математичних засобів для реалізації основного завдання: здійснення підтримки діяльності людини та пошуку інформації в режимі просунутого діалогу природною мовою [6].

1.2 Використання задач прогнозування у керуванні компанією

Прогнозування передбачає огляд того, що може статися в майбутньому, шляхом розгляду минулих і теперішніх подій та інцидентів. Прогнозування – це

інструмент прийняття рішень, який допомагає підприємствам справлятися з невизначеністю, що оточує бізнес, ретельно вивчаючи історичні дані та тенденції.

Прогнозування також можна назвати інструментом планування, який дозволяє підприємствам відповідно планувати майбутні кроки та бюджети. Компанії використовують цей інструмент з надією, що він охопить усі невизначеності, які можуть виникнути. Зазвичай вважається хорошою практикою вказувати ступінь невизначеності, пов’язаної з прогнозами. Треба завжди пам’ятати одне: дані мають бути актуальними, щоб спрогнозувати точно. Прогнозування базується на багатьох припущеннях, таких як [7-10]:

1) Повторюваність минулих подій, явищ, тенденцій тощо.

2) Зі скороченням горизонту прогнозу точність прогнозу зростає. Наприклад, прогноз на завтра буде більш точним, ніж прогноз на наступний місяць, а прогноз на наступний місяць буде точнішим, ніж прогноз на наступний рік.

3) Прогнозування в сукупності точніше, ніж прогнозування окремих елементів. Це означає, що компанія може спрогнозувати загальний попит на весь спектр товарів точніше, ніж вона може прогнозувати окремі складські одиниці. Наприклад, наша міжнародна компанія може більш точно спрогнозувати загальну кількість вчасно доставлених товарів з усіх складських блоків та способів доставки, ніж загальну кількість товарів доставлених у терміни літаком з складського блоку А.

4) Прогнози малоймовірно бувають точними. Тому доцільно запропонувати прогнозний «діапазон». Якщо передбачити попит на 100 000 одиниць на наступний місяць, вкрай ймовірно, що попит дорівнюватиме 100 000. Однак прогноз від 90 000 до 110 000 забезпечить набагато більшу ціль для планування.

Існує три основних типи прогнозування, незалежно від часового горизонту, які використовуються організаціями [11, с. 174].

Економічні прогнози стосуються ділового циклу. Вони прогнозують початок будівництва житла, рівень інфляції, грошову масу та інші показники [11, с. 175].

Технологічні прогнози відстежують темпи технічного прогресу. Це тримає організації в курсі тенденцій і може призвести до нових захоплюючих продуктів. Нові продукти можуть вимагати нових приміщень та обладнання, які необхідно планувати у відповідні часові рамки [11, с. 175].

Прогнози попиту стосуються продукції компанії та оцінюють споживчий попит. Вони також називаються прогнозами продажів, які мають багато цілей. На додаток до планування, виробництва та потужності, вони входять до фінансових, кадрових і маркетингових планів [11, с. 176].

Сучасні підприємства отримують дані про споживачів у реальному часі з багатьох джерел. За допомогою моделей прогнозування продажів можна легко оцінити майбутній попит на певний продукт або послугу, використовуючи дані споживачів у реальному часі [12, с. 21].

Інвестори використовують прогнозування, щоб вирішити, як можуть події вплинути на бізнес, наприклад очікувані продажі, завищення або зниження цін товарів конкретного бізнесу. Методи прогнозування також є важливим орієнтиром для підприємств, яким потрібна довгострокова перспектива діяльності за допомогою ключових історичних і нестаціонарних даних [12, с. 48].

Біржові аналітики використовують інструмент прогнозування для екстраполяції того, як тенденції, наприклад, ВВП або безробіття, можуть змінитися в наступному фінансовому кварталі чи році. Прогнозування на довший період зменшує шанси на точне прогнозування, оскільки може статися багато невизначених подій [13, с. 35].

Зрештою, статистики можуть використовувати прогнозування для аналізу ймовірного впливу змін у бізнес-операціях. Наприклад, історичні дані можуть бути зібрані для впливу на задоволеність клієнтів шляхом зміни способу доставки, або зміни в пріоритетах товарів можуть змінити загальну продуктивність компанії [14].

Методи прогнозування стосуються поставленої проблеми або набору даних. Аналітики роблять певні припущення щодо аналізованих обставин, які мають бути встановлені до визначення факторів прогнозування. Завдяки вирішеним питанням

вибирається відповідні атрибути з набору даних, які використовуються для інтерпретації інформації та подальшої роботи [14, с. 168].

Дані в цих методах прогнозування розбиваються на навчальну та тестову вибірки і відповідно визначається прогнозований результат. Потім відбувається період перевірки під час якого прогноз на тренувальних даних порівнюється з результатами на тестовій вибірці для створення більш точної моделі прогнозування [15, с. 104].

Чому прогнозування важливо? Компанія повинна постійно змінюватися, щоб задовольнити потреби клієнтів. Щоб зробити найбільш вигідні зміни, необхідно проаналізувати наявне та передбачити, що станеться в майбутньому. Тут вступає в дію бізнес-прогнозування. Прогнозування - це практика передбачення майбутніх подій [15, с. 105].

Існують два типи інструментів прогнозування. Якісні інструменти: вони засновані на судженні, яке ми робимо на основі досвіду та аналізу майбутніх тенденцій. Через залежність цього інструменту від індивідуальних суджень на прогноз впливають людські упередження. Кількісні інструменти: ці інструменти прогнозують дані шляхом аналізу минулих даних. Крім того, вони покладаються на статистичні методи для прогнозування. Ці методи можуть бути: метод експертних оцінок, аналіз часових рядів, казуальні (причино-наслідкові) методи прогнозування [16, с. 261].

Біржові аналітики використовують різні методи прогнозування, щоб визначити, як ціна акцій зміниться найближчим часом. Вони дивляться на дохід і порівнюють його з різними економічними показниками. Будь-які зміни фінансових або статистичних даних щохвилинно спостерігаються, щоб визначити зв’язок між різними змінними [17, с. 352].

Наприклад, прогноз продажів може базуватися на проміжку часу (період найближчих 12 місяців) або розвитку певних подій (купівля бізнесу конкурента). Нижче наведено два основні методи прогнозування. Ці два методи намагаються передбачити, що станеться в майбутньому [17, с. 355].

Якісний метод прогнозування: він також відомий як метод оцінки. Цей метод, здебільшого, дає суб’єктивні результати. Якісні прогнози, як правило, базуються на судженнях експертів або прогнозистів [17, с. 356].

Якісні прогнози найчастіше бувають упередженими, оскільки складаються зі знань, інтуїції та досвіду експерта. Вони нечасто базуються на даних, що робить процес нематематичним [17, с. 356].

Одним із прикладів такого способу прогнозування, який спадає на думку, є коли людина прогнозує результати продажу нового товару за особистим рейтингом. Це, звичайно, ґрунтується на особистій мотивації та зацікавленості. Найбільшою слабкістю цього методу є його неточність і високий відсоток невдач. Відсоток невдач високий, оскільки прогноз не базується на попередніх даних і фактах. Швидше вони базуються на особистих емоціях і мотиваціях [17, с. 357].

Метод кількісного прогнозування: цей метод абсолютно протилежний своєму еквівалентові. Це математичний процес, який робить його більш послідовним і об'єктивним. Кількісний метод не передбачає базування результатів на думці та інтуїції окремої людини. Натомість він використовує величезну кількість даних і фактів, доступних у минулому, і після їх аналізу робиться висновок [17, с. 358].

Наприклад, класичними прикладами кількісного методу прогнозування є моделі прогнозування, засновані на часових рядах, дисконтуванні, аналізі показників, які можуть випереджати чи відставати, а також економічне моделювання [17, с. 358].

Висновки до розділу 1

1. Інтелектуальні системи (ІС) – породили інші принципи організації компонентів систем, з'явилися інші поняття, терміни, блоки, які раніше не зустрічалися в розробках

2. ІС можуть вирішувати інтелектуальні завдання, розпізнавати ситуації (образи), навчатися поняттям і навичкам, формувати модель обстановки (розв'язуваної задачі), планувати поведінку (приймати рішення)

3. Основною властивістю природних ІС є їхня здатність до адаптації при зміні умов функціонування. Здатність до адаптації шляхом самоорганізації ґрунтується як на множинності елементів системи та розгалуженості зв'язків між ними.

4. Основними функціями системи, що самоорганізується, є функції інформаційного забезпечення, матеріального та енергетичного забезпечення, переміщення та адаптації.

5. Види інтелектуальних систем: експертні системи, гібридні інтелектуальні системи, інтелектуально–інформаційні системи

6. Прогнозування можна назвати інструментом планування, який дозволяє підприємствам відповідно планувати майбутні кроки та бюджети.

7. Існує три основних типи прогнозування, незалежно від часового горизонту, які використовуються організаціями: економічні, технологічні та прогнози попиту.

8. Методи прогнозування стосуються поставленої проблеми або набору даних. Аналітики роблять певні припущення щодо аналізованих обставин, які мають бути встановлені до визначення факторів прогнозування.

2 ВИБІР ТА ОПИС МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ЗАДАЧІ КЛАСИФІКАЦІЇ СЕРВІСІВ ДОСТАВКИ

Класифікація — це двоетапний процес у машинному навчанні, етап навчання та етап прогнозування. Алгоритми класифікації використовуються для класифікації нових спостережень у попередньо визначених класах для неініційованих даних. Модель буде використовувати набір навчальних даних і обчислить, як найкраще зіставити приклади вхідних даних із конкретними мітками класів. На етапі навчання модель розробляється на основі навчальних даних. На етапі прогнозування модель використовується для прогнозування значення для заданих даних.

2.1 Попередня обробка і аналіз даних

Попередня обробка даних – це процес отримання необроблених даних та перетворення їх у чисті, сформовані набори, які дозволяють проводити інтелектуальний аналіз, обробку та аналіз даних. Оскільки програміст слабо контролює збір даних або використовує різні вхідні дані, попередня обробка даних є необхідним кроком, оскільки необроблені дані зазвичай неповні або непослідовні у форматуванні. Правильне попереднє оброблення даних часто може вплинути на точність і адекватність прогнозування, зробивши його більш надійним і ретельним [18].

Дуже часто отримані дані, що використовуються в подальшому для обробки алгоритмами Data Mining (а також іншими методами обробки та аналізу даних) для вирішення прикладних завдань, мають погану якість. Для якісного перетворення потрібна первинна обробка, очищення даних вибірки [19].

Необроблені дані зазвичай мають наступні характеристики: дублювання, невідповідності, шуми, викиди, порожні або NaN значення, аномалії та безліч інших проблем. Отже, дані потрібно очищати, щоб отримати правильний та найбільш точний результат [19].

Очищення даних у статистичній вибірці - одне з найактуальніших завдань аналізу. На його виконання витрачається більшість часу при створенні рішень (іноді до 80% всього часу, відведеного на весь проект). Інструменти очищення даних не позбавляють користувача від роботи, деякі брудні дані взагалі не піддаються автоматичному очищенню через різнорідність та специфічність інформації. Крім того, слід зазначити, що відсутні універсальні способи усунення помилок даних вибірки і кожна ситуація, виходячи з складності, вимагає застосування свого методу або сукупності методів [20, с. 309].

Тільки у разі достатньо надійних та «чистих» значень даних можливе застосування алгоритмів Data Mining та методів машинного навчання для створення моделі та подальшої роботи з нею. Для покращення якості вихідної інформації доводиться використовувати усі можливі способи як організаційні, і програмні [20, с. 310].

Попередня обробка даних може допомогти досягти кращих результатів [21]:

- Підвищення точності. Видаляючи відсутні або неузгоджені значення даних, викликані людськими або комп'ютерними помилками, точність набору даних підвищується.

- Підвищення консистенції. Можуть виникати дублікати даних і їхнє видалення під час попередньої обробки допомагає забезпечити аналіз більш узгоджених значень даних та отримання надійних результатів, які не будуть спотворені.

- Дані стають повнішими. Попередня обробка даних дозволяє додавати відсутні дані там, де це необхідно.

- Оброблені дані полегшують роботу обраного алгоритму. Попередня обробка зазвичай спрощує читання, використання та інтерпретацію даних, особливо під час використання автоматизованого програмного забезпечення для машинного навчання.

Двома основними функціями попередньої обробки даних є перевірка даних та редагування даних. Перевірка даних: перевірка даних оцінює, чи є дані для

проекту повними та точними, щоб згодом отримати найкращі результати. Редагування даних: введення відсутніх значень або виправлення помилок у даних, які можуть бути виявлені в процесі перевірки.

П’ять кроків для попередньої обробки даних [20, с. 289-297]:

1. Оцінка даних. Проведення оцінки якості даних допомагає встановити, наскільки достовірні дані, і на цьому етапі зазвичай виконується як перевірку даних, так і заміна помилкових даних, це можуть бути:

- Змішані значення даних: збір даних з різних джерел часто призводить до унікальних значень даних, наприклад, наявність у наборі даних про купівлі різних товарів тільки чоловічої статі. Необхідно позначити ці дані, вибрати, яке значення присвоїти замість обох, та змінити будь-які відповідні дані на наступному етапі попередньої обробки.

- Невідповідність даних: під час збору даних поширені різні числові формати даних. Наприклад, можна побачити ціле число без десяткових знаків або формати з плаваючою комою з десятковими знаками.

- Різні масиви даних. Об'єднання агрегованих даних з окремих наборів даних часто означає, що деякі набори містять поля, яких немає в інших.

- Викиди даних. Екстремальні викиди у даних можуть вплинути на результати, особливо під час автоматичного аналізу за допомогою машинного навчання. Потрібно переглянути будь-які викиди, щоб з'ясувати, чи є вони законними і чи повинні вони бути частиною обробки даних або були помилкою, допущеною під час збору даних.

2. Очищення даних. Після оцінки даних відбувається їх обробка на основі результатів, отриманих на першому етапі. Очищення даних спрямовано створення простих і повних наборів даних для програм, що дозволяють виконувати аналіз. Дві поширені причини, з яких відбувається очищення даних, включають:

- Відсутні дані: це може статися через людську помилку, збої у роботі програми або інших факторів, а заміна відсутніх даних допомагає забезпечити точність та надійність майбутнього аналізу.

- Зашумлені дані: дані, які не мають значущої цінності, є шумом, наприклад, дублюючі вхідні дані або поля даних, що не відносяться до аналізу.

При виявленні відсутніх даних у наборах даних відбувається видалення стовпців, рядків та полів перед об'єднанням даних. Щоб вирішити проблему зашумлених даних, існують такі варіанти, щоб переконатися, що алгоритм зрештою зможе зчитати дані:

- Регресія. Адаптація даних під функції множинної або лінійної регресії, особливо корисна за наявності великого набору даних.

- Бінарізація: поділ даних на згладжені сегменти або комірки однакового розміру, наприклад, коли є набір даних вікового діапазону. Можна згрупувати дані за групами категорій, наприклад, за віком 21–39 років, 40–58 років та 58–76 років.

- Кластеризація: групування даних у кластери схожих даних з урахуванням екстремальних викидів за спільними ознаками.

3. Інтеграція та перетворення даних. На цьому етапі відбувається інтеграція різних наборів даних після їх повного очищення. Хоча дані вже змінені, цей крок відбувається для подальшого перетворення даних у належні формати, які комп'ютерне програмне забезпечення та машинне навчання можуть читати та інтерпретувати. Існує безліч способів перетворення даних, у тому числі:

- Агрегація: цей процес поєднує дані, файли та записи для зменшення загального обсягу. Наприклад, замість цього можна об'єднувати тисячі щоденних бізнес-транзакцій у тижневі чи місячні значення.

- Нормалізація: цей процес перевіряє дані, щоб переконатися, що вони зберігаються в одному місці і немає надмірності та в одному форматі.

- Дискретизація: необроблені значення замінюються рівнями інтервалів шляхом розподілу діапазону інтервалів атрибутів. Наприклад, використовуючи терміни "підліток", "середній вік" або "старший" замість числових значень віку або кластерів.

- Узагальнення: цей метод можна використовувати для переміщення точок даних нижчого рівня до точок даних вищого рівня в залежності від цілей аналізу.

Наприклад, дані, які містять домашні адреси, назви вулиць та поштові індекси, можуть бути узагальнені та переміщені у категорії високого рівня, такі як міста, округи, регіони чи штати.

4. Скорочення даних. Великі набори даних іноді можуть зробити базу даних повільною, дорогою та складною для зберігання та отримання даних. Натомість доцільно виконати скорочення даних, щоб мати менше представлень даних у базі даних, зазвичай використовуючи методи кодування. Деякі методи, які використовуються під час перетворення даних, застосовні і до скорочення. Ось ще кілька варіантів:

- Вибір атрибута: об'єднання нових та існуючих функцій у наборі даних для більш ефективного аналізу називається вибором атрибута. Наприклад, можна додати «студент» у поля для чоловіків та жінок, щоб проаналізувати, скільки чоловіків та жінок є студентами, незалежно від їх конкретних областей навчання.

- Зменшення розмірності: коли набори даних, пов'язані з реальними задачами, потребують якісного аналізу, а не швидкості, наприклад, комп'ютерного зору, перекладу або генерації мови.

- Зменшення кількості: цей процес замінює вихідні дані на менші форми, які служать для подання з використанням параметричних або непараметричних методів. Параметричний підхід використовує моделі, які зазвичай генеруються за допомогою регресії, тоді як непараметричні методи використовують вибірку даних, агрегацію куба даних і гістограми.

5. Зразок даних. Залежно від ситуації дані можуть бути простими або складнішими для роботи, альтернативним варіантом може бути вибірка даних. Наприклад, у програміста можуть бути обмеження пам'яті, сховища або часу при роботі з великими наборами даних, і натомість він може використовувати частину набору для проведення аналізу. Вибірка даних часто дає однакові результати, якщо підмножина даних має самі властивості, як і вихідна.

2.2 Вибір базових класифікаторів

У машинному навчанні класифікація відноситься до задачі прогнозного моделювання, коли мітка класу прогнозується для вхідних даних. Для класифікації потрібен навчальний набір із безліччю записів для навчання.

Модель буде використовувати набір навчальних даних і обчислить, як найкраще зіставити приклади вхідних даних із конкретними мітками класів. Алгоритми класифікації дозволяють побудувати початкову (базову) модель, яка відноситиме кожне нове спостереження до певного класу. У якості базових моделей обрано моделі: дерева рішень (Decision Tree), Наївний Баєсів класифікатор (NB), лінійний дискримінантний аналіз (LDA), квадратичний дискримінантний аналіз (QDA), логістична регресія (LR), метод опорних векторів (SVM), метод найближчого сусіда (KNN), штучні нейронні мережі (ANN) та модель випадкового лісу (RF).

- Алгоритм дерева рішень (Decision Tree)

Дерево рішень є одним із найпростіших і найпопулярніших алгоритмів класифікації для розуміння та інтерпретації. Цей алгоритм належить до сімейства алгоритмів навчання з вчителем. На відміну від інших алгоритмів навчання з вчителем, алгоритм дерева рішень також можна використовувати для вирішення проблем регресії та класифікації [22].

Метою використання дерева рішень є створення навчальної моделі, яка може використовуватися для прогнозування класу або значення цільової змінної шляхом вивчення простих правил прийняття рішень, виведених із попередніх даних (навчальних даних) [23, с. 208].

У деревах рішень прогнозування мітки класу для запису починається з кореня дерева. Ми порівнюємо значення кореневого атрибута з атрибутом запису. На основі порівняння ми слідуємо за гілкою, що відповідає цьому значенню, і переходимо до наступного вузла [23, с. 211].

Типи дерев рішень базуються на типі цільової змінної, яку ми маємо. Він може бути двох видів [24]:

Дерево рішень категоріальної змінної: Дерево рішень, яке має категоріальну цільову змінну, а потім називається деревом рішень категоріальної змінної.

Дерево рішень безперервної змінної: Дерево рішень має безперервну цільову змінну, тоді воно називається деревом рішень безперервної змінної.

Наприклад, у нас є проблема передбачити, чи сплатить клієнт страховій компанії премію за поновлення (так/ні). Тут ми знаємо, що дохід клієнтів є важливою змінною, але страхова компанія не має відомостей про доходи для всіх клієнтів. Оскільки ми знаємо, що це важлива змінна, ми можемо побудувати дерево рішень, щоб передбачити дохід клієнта на основі професії, його витрат та інших змінних. У цьому випадку ми прогнозуємо значення безперервних змінних [8].

Кожен вузол у дереві діє як тестовий приклад для певного атрибута, і кожне ребро, що йде від вузла, відповідає можливим відповідям на тестовий приклад. Цей процес є рекурсивним за своєю природою і повторюється для кожного піддерева з коренем у новому вузлі.

Основна проблема в реалізації дерев рішень полягає в тому, щоб визначити, які атрибути ми повинні розглядати як кореневий вузол і на кожному рівні. Рішення про стратегічні поділи сильно впливає на точність дерева. Критерії прийняття рішень різні для дерев класифікації та регресії [24].

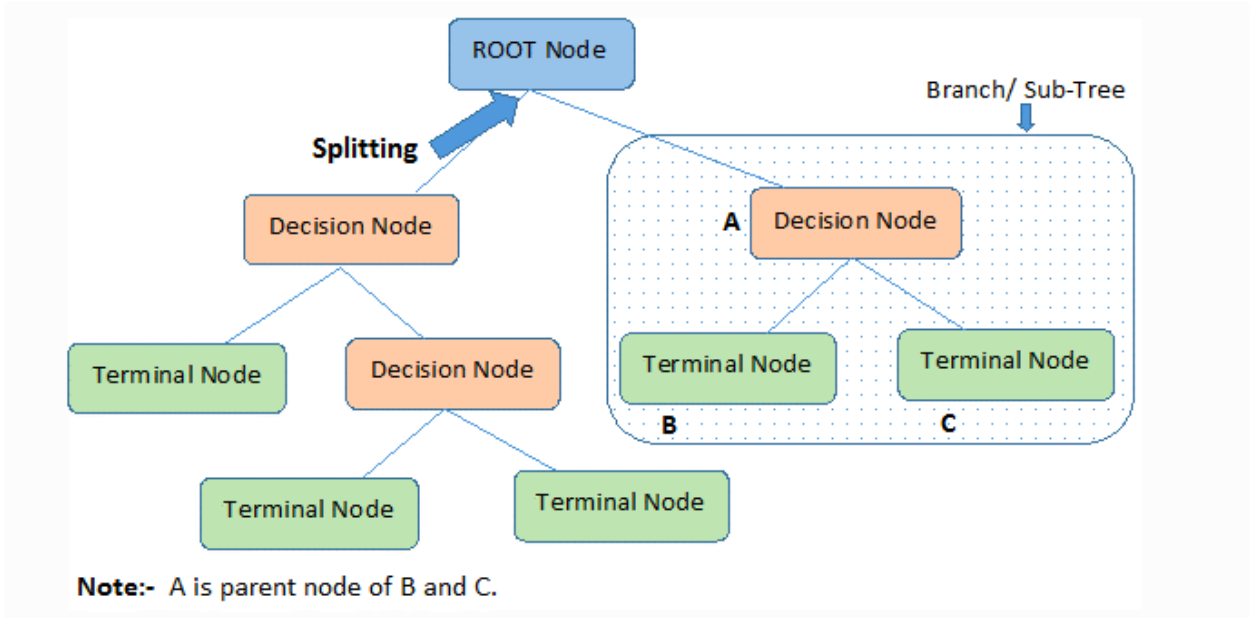


Рисунок 2.1 – Структура дерева рішень

Дерева рішень використовують кілька алгоритмів, щоб вирішити, чи розділити вузол на два або більше підвузлів. Створення підвузлів збільшує однорідність результуючих підвузлів. Іншими словами, можна сказати, що чистота вузла збільшується щодо цільової змінної. Дерево рішень розбиває вузли на всі доступні змінні, а потім вибирає поділ, який призводить до найбільш однорідних підвузлів [24].

Вибір алгоритму також базується на типі цільових змінних. Розглянемо деякі алгоритми, які використовуються в деревах рішень [25 с. 521]:

ID3 → (розширення D3)

C4.5 → (наступник ID3)

CART → (Дерево класифікації та регресії)

CHAID → (Автоматичне виявлення взаємодії хі-квадрат. Виконує багаторівневе розбиття під час обчислення дерев класифікації)

MARS → (сплайни багатовимірної адаптивної регресії)

Алгоритм ID3 будує дерева рішень, використовуючи підхід жадібного пошуку зверху вниз у просторі можливих гілок без повернення назад. Жадібний алгоритм, як випливає з назви, завжди робить вибір, який здається найкращим на даний момент. [25, с. 522]

- Наївний Баєсів класифікатор (NB)

Наївний алгоритм Байєса є одним із важливих алгоритмів у машинному навчанні, який допомагає вирішити проблеми класифікації. Він походить від теорії ймовірностей Байєса та використовується для класифікації, де навчаються багатовимірні набори даних.

Наївний алгоритм Байєса відомий своєю простотою та ефективністю. За допомогою цього алгоритму швидше будувати моделі та робити прогнози. При створенні будь-якої моделі ML краще застосовувати теорему Байєса [26].

Це алгоритм, який вивчає ймовірність кожного об’єкта, його особливості та групи, до яких вони належать. Він також відомий як імовірнісний класифікатор.

Наївний алгоритм Байєса підпадає під навчання з вчителем та в основному використовується для вирішення проблем класифікації [26].

Наприклад, ви не можете ідентифікувати автомобіль за його ознаками та кольором, оскільки існує багато авто із подібними ознаками. Але робиться імовірнісне передбачення приблизно того ж, і ось тут буде корисним наївний алгоритм Байєса.

Імовірність є основою для алгоритму Наївного Байєса. Цей алгоритм побудовано на основі ймовірнісних результатів, які він може запропонувати для нерозв'язних проблем за допомогою передбачення. Імовірність допомагає передбачити настання події з усіх потенційних результатів. Математичне рівняння ймовірності таке: *Імовірність події = кількість сприятливих подій / загальна кількість результатів, де 0 <= ймовірність події <= 1*. Сприятливий результат означає подію, яка є результатом ймовірності. Імовірність завжди становить від 0 до 1, де 0 означає відсутність ймовірності того, що це відбудеться, а 1 означає ймовірність успіху цієї події [27, с. 448].

Для кращого розуміння ви також можете розглянути випадок, коли ви передбачите товар від оригінального виробника на основі його відгуків та ціни від постачальника. Ось кілька можливих припущень, які можна зробити. Можливим є обрання оригінального товару, який необхідно замовити зі складу, або купівлі підробки низької якості та, власне, отримання помилкового результату. У будь-якому випадку ймовірність вибрати потрібний товар становить 50%.

Теорія Байєса працює над тим, щоб прийти до гіпотези (H) на основі заданого набору доказів (E). Це стосується двох речей: ймовірності гіпотези до доказів P(H) і ймовірності після доказів P(H|E). Теорія Байєса пояснюється наступним рівнянням [28, с. 165]:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} \qquad (2.1)$$

У наведеному вище рівнянні P(H|E) означає, як відбувається подія H, коли відбувається подія E; P(E|H) показує, як часто відбувається подія E, коли подія H

відбувається першою; P(H) представляє ймовірність того, що подія X відбудеться сама по собі; P(E) представляє ймовірність того, що подія Y відбудеться сама по собі. Правило Байєса – це метод визначення P(H|E) з P(E|H). Коротше кажучи, він надає спосіб обчислення ймовірності гіпотези з наданими даними [28, с. 165].

Умовна ймовірність є підмножиною ймовірності. Коли ви берете події X і Y, умовна ймовірність події Y визначається як ймовірність того, що подія відбудеться, коли подія X уже закінчилася. Він записується як P(Y|X). Математична формула для цього така [29, с. 154]:

$$P(Y|A) = \frac{P(X\ and\ Y)}{P(X)} \qquad (2.2)$$

Байєсова ймовірність дозволяє розрахувати умовні ймовірності. Це дозволяє використовувати часткові знання для розрахунку ймовірності настання конкретної події. Важливо те, що ви не можете точно визначити, як докази вплинуть на ймовірність події, але ви можете визначити точну ймовірність [29, с. 155].

Цей алгоритм використовується для розробки моделей для задач прогнозування та класифікації. Є навчальні дані, щоб навчити модель і зробити її функціональною. Потім потрібно перевірити дані для оцінки моделі та створення нових прогнозів. Нарешті, потрібно визначити вхідні атрибути «певними ознаками» і позначити їх «виходами» у навчальних даних [30].

Використовуючи умовну ймовірність, позначену як P(E|O), можливо обчислити ймовірність доказів на основі даних виходів. Кінцева мета — обчислити P(O|E) — ймовірність результату на основі поточних атрибутів. Коли задача має два виходи, можна обчислити ймовірність кожного результату та сказати, який з них переможе [30].

- Лінійний дискримінантний аналіз (LDA)

Лінійний дискримінантний аналіз (англ. Linear Discriminant Analysis, LDA) є узагальненням лінійного дискримінанта Фішера, методу, що використовується розпізнавання образів та навчання машин для пошуку лінійної комбінації ознак, яка описує чи поділяє два або більше класів чи подій. Комбінація, що вийшла, може

бути використана як лінійний класифікатор, або, більш часто, для зниження розмірності перед класифікацією [31, с. 87].

LDA тісно пов'язаний з дисперсійним аналізом (англ. ANalyse Of Variance=ANOVA) і регресійним аналізом, які намагаються виразити одну залежну змінну як лінійну комбінацію інших ознак чи вимірів. Проте дисперсійний аналіз використовує якісні незалежні змінні та безперервну залежну змінну, у той час як дискримінантний аналіз має безперервні незалежні змінні та якісну залежну змінну (тобто мітку класу). Логістична регресія та пробіт-регресія більше схожі на LDA, ніж дисперсійний аналіз, оскільки вони так само пояснюють якісну змінну через безперервні незалежні змінні. Ці інші методи кращі в застосунках, в яких немає здогадів, що незалежні змінні нормально розподілені, що є фундаментальним припущенням методу LDA [31, с. 48].

Лінійний дискримінантний аналіз тісно пов'язаний також з методом головних компонентів (англ. Principal Component Analysis, PCA) і факторним аналізом тим, що вони шукають лінійні комбінації змінних, які краще пояснюють дані. LDA явно намагається моделювати різницю між класами даних. Метод головних компонентів, з іншого боку, не бере до уваги будь-яку різницю в класах, а факторний аналіз будує комбінації ознак, спираючись швидше на відмінності, а не на подібність. Дискримінантний аналіз відрізняється також від факторного аналізу тим, що не є незалежною технікою — для його роботи має бути визначена різниця між незалежними змінними та залежними змінними (останні також називаються критеріальними змінними) [32].

Лінійний дискримінантний аналіз працює, коли виміри, зроблені на незалежних змінних кожного спостереження, є безперервними величинами. Коли маємо справу з якісними незалежними змінними, еквівалентною технікою є дискримінантний аналіз відповідностей [32].

LDA зосереджується головним чином на проектуванні функцій у просторі вищих вимірів у нижчі виміри. Ви можете досягти цього в три кроки [31, с. 56]:

- По-перше, потрібно обчислити роздільність між класами, яка є відстанню між середнім значенням різних класів. Це називається міжкласовою дисперсією.

$$S_b = \sum_{i=1}^{g} N_i(\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T \qquad (2.3)$$

- По-друге, потрібно обчислити відстань між середнім значенням і вибіркою кожного класу. Її також називають внутрішньокласовою дисперсією.

$$S_b = \sum_{i=1}^{g} N_i(\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x_j})(x_{i,j} - \bar{x_j})^T \qquad (2.4)$$

- Потрібно побудувати простір нижчих розмірів, який максимізує дисперсію між класами та мінімізує дисперсію всередині класу. P розглядається як проекція простору нижчої вимірності, яка також називається критерієм Фішера.

$$P_{lda} = arg \max_P \frac{|P^T S_b P|}{|P^T S_w P|} \qquad (2.5)$$

Лінійний дискримінантний аналіз використовується, коли групи відомі апріорі (на відміну кластерного аналізу). Висловлюючись простими термінами, аналіз дискримінантних функцій є класифікацією, що розбиває об'єкти на групи, класи чи категорії певного типу. Лінійний дискримінантний аналіз є простим і ефективним методом класифікації. Оскільки він простий і добре зрозумілий, існує багато розширень і варіацій методу. Серед популярних розширень [32]:

- Квадратний дискримінантний аналіз (QDA): кожен клас використовує власну оцінку дисперсії (або коваріації, якщо є кілька вхідних змінних).

- Гнучкий дискримінантний аналіз (FDA): де використовуються нелінійні комбінації вхідних даних, наприклад сплайни.

Регуляризований дискримінантний аналіз (RDA): вводить регуляризацію в оцінку дисперсії (фактично коваріації), пом’якшуючи вплив різних змінних на LDA [34].

- Квадратичний дискримінантний аналіз (QDA)

Квадратичний дискримінантний аналіз (QDA) тісно пов’язаний з лінійним дискримінантним аналізом (LDA), де передбачається, що дані розподілені

нормально. Проте, на відміну від LDA, у QDA немає припущення, що коваріація кожного з класів є ідентичною. Щоб оцінити параметри, необхідні для квадратичної дискримінації, потрібно більше обчислень і даних, ніж у випадку лінійної дискримінації. Якщо немає великої різниці в групових коваріаційних матрицях, то остання працюватиме так само добре, як і квадратична дискримінація. Квадратична дискримінація є загальною формою Байєсівської дискримінації [31, с. 74].

Цей метод схожий на LDA і також передбачає, що спостереження з кожного класу розподіляються нормально, але LDA не передбачає, що кожен клас має однакову коваріаційну матрицю. Натомість QDA передбачає, що кожен клас має власну коваріаційну матрицю. Тобто передбачається, що спостереження з k класу має форму $X \sim N(\mu_k, \Sigma_k)$ [34].

Використовуючи це припущення, QDA потім знаходить такі значення [34]: μ_k – середнє значення всіх спостережень за навчанням з k-го класу, Σ_k – коваріаційна матриця k-го класу, π_k – частка навчальних спостережень, які належать до k-го класу.

Потім QDA підключає ці числа до наступної формули та призначає кожне спостереження X = x класу, для якого формула дає найбільше значення:

$$D_k(x) = -\frac{1}{2} \cdot (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \cdot \log|\Sigma_k| + \log|\pi_k| \qquad (2.6)$$

Зауважимо, що QDA має «квадратична» у своїй назві, оскільки значення, створене функцією вище, походить від квадратичних функцій x.

Дискримінантний аналіз використовується для визначення того, як змінні розподіляються на дві або більше природних груп. Наприклад, дослідник освітнього процесу може захотіти дослідити, на які змінні розрізняють випускників середньої школи, які вирішили (1) вступати до коледжу, (2) НЕ вступати до коледжу. З цією метою дослідник міг би зібрати дані про численні змінні до випуску студентів. Після закінчення школи більшість студентів, природно, потраплять до однієї з двох категорій. Потім можна використати дискримінантний

аналіз, щоб визначити, яка(і) змінна(і) є найкращим предиктором подальшого вибору освіти студентами [28, с. 129].

Обчислювальний аналіз дискримінантної функції дуже схожий на дисперсійний аналіз (ANOVA). Наприклад, припустимо той самий сценарій випускника школи. Ми могли б досліджувати заявлений намір студентів продовжувати навчання в університеті за рік до закінчення навчання. Основна ідея, що лежить в основі дискримінантного аналізу, полягає в тому, щоб визначити, чи відрізняються групи за середнім значенням змінної, а потім використовувати цю змінну для прогнозування належності до групи (наприклад, нових випадків) [18].

Дискримінантний аналіз можна використовувати для досягнення двох цілей: якщо ми хочемо оцінити адекватність класифікації, враховуючи групову приналежність досліджуваних об’єктів; або ми хочемо визначити об'єкти до однієї з кількох (відомих) груп об'єктів. Таким чином, дискримінантний аналіз може мати описову або прогнозну мету. В обох випадках перед виконанням дискримінантного аналізу необхідно знати деякі характеристики груп. Отже, дискримінантний аналіз можна використовувати як корисне доповнення до кластерного аналізу (щоб оцінити результати останнього) або аналізу основних компонентів [35, с. 64].

- Логістична регресія (LR)

Логістична регресія – це алгоритм навчання з вчителем, який використовується для прогнозування залежної категоріальної цільової змінної. По суті, якщо у вас є великий набір даних, які ви хочете класифікувати, логістична регресія може допомогти [36, с. 103].

Наприклад, якщо є собака й апельсин й потрібно з’ясувати, чи є кожен із цих предметів твариною чи ні, бажаним результатом було б, щоб собака була класифікована як тварина, а апельсин – бути віднесеним до категорії не тварин. У цьому прикладі є лише дві можливі відповіді (двійкова логістична регресія), тварина чи не тварина. Однак також можна налаштувати свою логістичну регресію з більш ніж двома можливими категоріями (мультиноміальна логістична регресія).

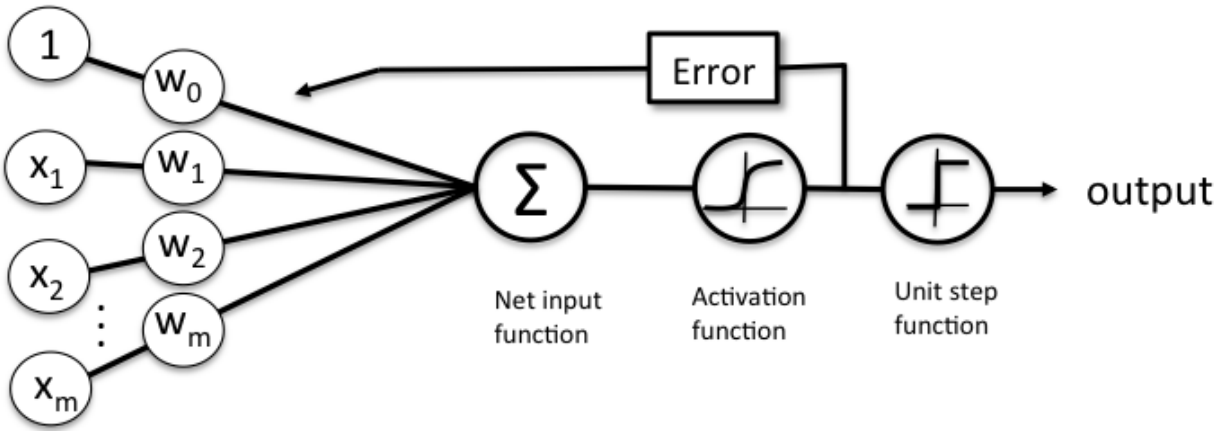


Рисунок 2.2 – Схема класифікатору логістичної регресії

Щоб глибше зануритися в те, як ваша модель може спробувати класифікувати ці два елементи безпосередньо, давайте розглянемо, що ще потрібно знати моделі про елементи, щоб вирішити, куди вони належать. При розгляді того, як класифікувати кожен елемент або точку даних, слід розглянути інші подібні аспекти цих елементів. Аспекти або особливості можуть включати колір, розмір, вагу, форму, висоту, об’єм або кількість кінцівок. Таким чином, знання про те, що апельсин має форму кола, може допомогти алгоритму зробити висновок, що апельсин не був твариною. Подібним чином допомогло б і знання того, що в апельсина немає кінцівок.

Логістична регресія вимагає, щоб залежна змінна, в даному випадку чи був елемент тварина чи ні, була категоріальною. Результат або тварина, або не тварина — між ними немає діапазону. Проблема, яка має безперервний результат, наприклад прогнозування оцінки учня або запасу паливного бака автомобіля, не є гарним прикладом для використання логістичної регресії [37].

- Метод опорних векторів (SVM)

Метод опорних векторів або SVM є одним із найпопулярніших алгоритмів навчання з вчителем, який використовується для задач класифікації та регресії. Однак, в основному, він використовується для проблем класифікації в машинному навчанні.

Метою алгоритму SVM є створення найкращої лінії або межі рішення, яка може розділити n-вимірний простір на класи, щоб можна було легко помістити нову точку даних у правильну категорію в майбутньому. Ця межа найкращого рішення називається гіперплощиною [38].

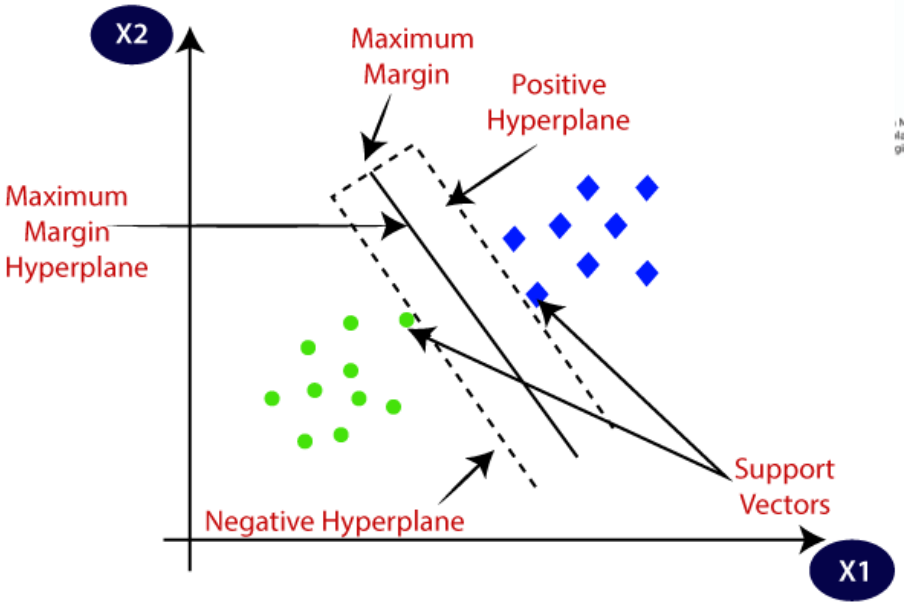


Рисунок 2.3 – Схема класифікатору опорних векторів

SVM вибирає крайні точки/вектори, які допомагають створити гіперплощину. Ці екстремальні випадки називають опорними векторами, а отже, алгоритм називають машиною опорних векторів. На рисунку 2.3 є дві різні категорії, класифіковані за допомогою границі рішення або гіперплощини [36, с.58].

Алгоритм SVM можна використовувати для розпізнавання обличчя, класифікації зображень, категоризації тексту тощо.

Метод опорних векторів може бути двох типів: *лінійний SVM* використовується для лінійно розділених даних, що означає, що якщо набір даних можна класифікувати на два класи за допомогою однієї прямої лінії, то такі дані називаються лінійно роздільними даними, а класифікатор використовується як лінійний класифікатор SVM; та *нелінійний SVM* використовується для нелінійно розділених даних, що означає, що якщо набір даних не можна класифікувати за допомогою прямої лінії, такі дані називаються нелінійними даними, а використовуваний класифікатор називається нелінійним [38].

- Метод найближчого сусіда (KNN)

Алгоритм k-найближчих сусідів (KNN) — це простий, легкий у реалізації алгоритм машинного навчання з вчителем, який можна використовувати для вирішення як проблем класифікації, так і регресії.

Алгоритм KNN припускає, що подібні речі існують у безпосередній близькості. Іншими словами, схожі речі знаходяться поруч один з одним.

Розглянемо нижче наведений рисунок, на якому показано, як подібні точки даних зазвичай знаходяться поблизу одна від одної. KNN фіксує ознаки подібності (іноді їх називають відстанню, близькістю) та обчислює відстані між точками на графіку [39, с. 21].

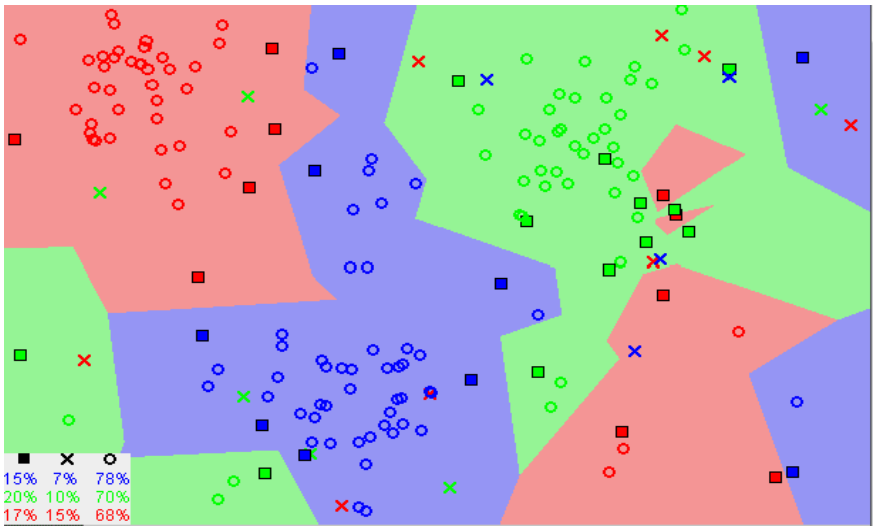


Рисунок 2.4 – Схема класифікатору k-найближчих сусідів

Існують інші способи обчислення відстані, і один спосіб може бути кращим залежно від проблеми, яку ми вирішуємо. Однак відстань по прямій лінії (також звана евклідовою відстанню) є популярним і знайомим вибором [39, с. 21].

Щоб вибрати *k*, який підходить для ваших даних, кілька разів запускаємо алгоритм KNN із різними значеннями *k* і вибираємо *k*, який зменшує кількість помилок, з якими ми стикаємося, зберігаючи здатність алгоритму точно робити прогнози, коли йому надаються дані, які він не має [39, с. 23].

Зменшуючи значення k до 1, прогнози стають менш стабільними. І навпаки, коли ми збільшуємо значення *k*, наші прогнози стають більш стабільними завдяки голосуванню більшості (усередненню) і, таким чином, більша ймовірність зробити

точніші прогнози (до певного моменту). Згодом ми починається збільшення кількості помилок [39, с. 24].

Головний недолік KNN, який полягає в тому, що він стає значно повільнішим із збільшенням обсягу даних, що робить його непрактичним вибором у середовищах, де прогнозування потрібно робити швидко. Крім того, існують швидші алгоритми, які можуть давати більш точні результати класифікації та регресії [27, с. 384].

Однак за умови, що у вас є достатні обчислювальні ресурси для швидкої обробки даних, які ви використовуєте для прогнозування, KNN може бути корисним у вирішенні проблем, які мають рішення, які залежать від ідентифікації подібних об’єктів. Прикладом цього є використання алгоритму KNN у системах рекомендацій, застосування KNN-пошуку [23, с. 76].

- Штучні нейронні мережі (ANN)

Штучні нейронні мережі (ANN) є одним з основних інструментів, які використовуються в машинному навчанні. Як випливає з «нейронної» частини їх назви, це системи, які моделюють роботу мозку та призначені для повторення того, як ми, люди, навчаємося.

Нейронні мережі складаються з вхідного та вихідного рівнів, а також (у більшості випадків) прихованого шару, що складається з одиниць, які перетворюють вхідні дані у те, що може використовувати вихідний рівень [40, с.23].

ANN мають три рівні, які взаємопов’язані. Перший шар складається з вхідних нейронів. Ці нейрони надсилають дані на другий рівень, який, у свою чергу, надсилає вихідні нейрони на третій рівень. ANN вважаються інструментами моделювання нелінійних статистичних даних, де моделюються складні зв’язки між входами та виходами або виявляються закономірності. Зверніть увагу, що нейрон також можна назвати перцептроном [41].

На рисунку 2.5 зображено приклад штучної нейронної мережі.

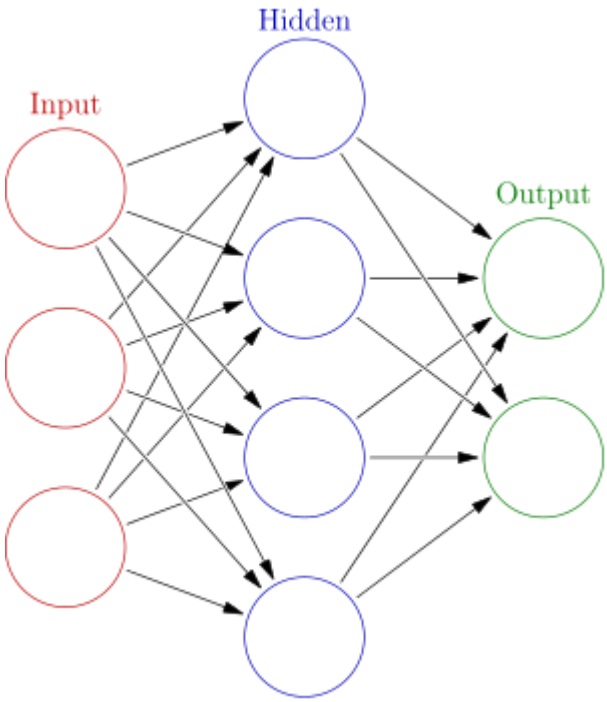


Рисунок 2.5 – Схема класифікатору на основі штучних нейронних мереж

Щоб отримати базове уявлення про те, як навчається нейронна мережа глибокого навчання, уявіть фабричну лінію. Після того, як вихідні матеріали (набір даних) введені, вони потім передаються конвеєрною стрічкою, з кожною наступною зупинкою або шаром, витягуючи інший набір характеристик високого рівня. Якщо мережа призначена для розпізнавання об’єкта, перший рівень може аналізувати яскравість його пікселів. Наступний шар може визначити будь-які краї зображення на основі ліній подібних пікселів. Після цього інший шар може розпізнавати текстури та форми тощо. До моменту досягнення четвертого чи п’ятого рівня мережа глибокого навчання створить складні детектори ознак. Вона може визначити, що певні елементи зображення (наприклад, пара очей, ніс і рот) зазвичай зустрічаються разом [42].

Існує кілька типів нейронних мереж, кожна з яких має свої специфічні варіанти використання та рівні складності. Найпростіший тип нейронної мережі — це нейронна мережа прямого зв’язку, у якій інформація рухається лише в одному напрямку від входу до виходу. Більш широко використовуваним типом мережі є рекурентна нейронна мережа, у якій дані можуть надходити в кількох напрямках. Ці нейронні мережі мають більші здібності до навчання та широко

використовуються для більш складних завдань, таких як вивчення рукописного тексту або розпізнавання мови [40, с. 47].

Існують також згорткові нейронні мережі, мережі машин Больцмана, мережі Хопфілда та багато інших. Вибір правильної мережі залежить від даних і конкретної програми. У деяких випадках може бути бажано використовувати кілька підходів, як, наприклад, у випадку складної задачі типу розпізнавання голосу [41].

Робота ANN бере свій початок від нейронної мережі, яка знаходиться в мозку людини. ANN працює на так званому прихованому стані. Ці приховані стани схожі на нейрони. Кожен із цих прихованих станів є тимчасовою формою, яка має ймовірнісну поведінку. Сітка такого прихованого стану діє як міст між входом і виходом [40, с. 35].

У нас є вхідний рівень, який є даними, які ми надаємо в нейронну мережу, є приховані шари та є вихідний рівень, де розміщуються готові обчислення мережі для використання [42].

Спочатку ваги мережі можуть бути випадковими. Коли вхідні дані передаються вхідному шару, процес просувається вперед і прихований рівень отримує вхідні дані разом із ваговими коефіцієнтами. Цей процес триває до тих пір, поки не буде досягнуто останнього рівня результату і не буде отримано результат. Коли отримано результат, він порівнюється з фактичним значенням та починає роботу алгоритм зворотного розповсюдження, щоб налаштувати ваги мережевих зв’язків для кращого результату. Що тоді роблять нейрони в шарах? Вони відповідають за індивідуальне навчання. Вони складаються з функції активації, яка дозволяє сигналу проходити чи ні залежно від того, яка функція активації використовується та який вхід надійшов із попереднього рівня. Зараз ми детально розглянемо функції активації [40, с. 26].

Функції активації дуже важливі для штучної нейронної мережі, щоб навчити та зрозуміти складне функціональне відображення між входами та змінною відповіді. Вони вводять нелінійні властивості в мережу. Їх основна мета —

перетворити вхідний сигнал вузла в ANN до вихідного сигналу. Цей вихідний сигнал тепер використовується як вхід на наступному рівні в стеку [40, с. 26].

Зокрема, в ANN ми створюємо суму добутків вхідних даних *(X)* та їхніх відповідних вагових коефіцієнтів *(W)* і застосовуємо до цього функцію активації *f(x)*, щоб отримати вихідні дані цього шару та передати їх як вхідні дані наступному шар [40, с. 27].

Найбільш популярні типи функцій активації – це сигмоїда або логістична, Tanh – гіперболічний тангенс та ReLu – випрямлений лінійний вузол [40, с. 27].

- Модель випадкового лісу (RF)

Випадковий ліс — це широко використовуваний алгоритм машинного навчання, який поєднує результати кількох дерев рішень для досягнення єдиного результату. Його простота використання та гнучкість сприяли його прийняттю, оскільки він обробляє як задачі класифікації, так і регресії [43].

Алгоритм випадкового лісу є розширенням беггінгу, оскільки він використовує як беггінг, так і випадковість ознак для створення некорельованого лісу дерев рішень. Випадковість функцій, також відома як бутстрепова агрегація або «метод випадкового підпростору», генерує випадкову підмножину функцій, що забезпечує низьку кореляцію між деревами рішень. Це ключова відмінність між деревами рішень і випадковим лісом. У той час як дерева рішень враховують усі можливі розподіли функцій, випадковий ліс вибирає лише підмножину цих функцій [17, с. 357].

Алгоритм випадкового лісу має три основні гіперпараметри, які необхідно встановити перед навчанням. До них належать розмір вузла, кількість дерев і кількість відібраних функцій. Саме тому класифікатор випадкового лісу можна використовувати для вирішення проблем регресії або класифікації [43].

З рисунку 2.6 видно, що алгоритм випадкового лісу складається з набору дерев рішень, і кожне дерево в ансамблі складається із вибірки даних, отриманої з навчального набору із заміною, яка називається початковою вибіркою. З цієї навчальної вибірки одна третина відкладена як тестові дані, відома як вибірка поза

пакетом (OOB – частота помилок out-of-bag). Інший випадок випадковості потім вводиться через беггінг, додаючи більше різноманітності до набору даних і зменшуючи кореляцію між деревами рішень. Залежно від типу проблеми визначення прогнозу буде різним. Для завдання регресії буде усереднено окремі дерева рішень, а для завдання класифікації – більшість голосів, тобто найпоширеніша категоріальна змінна — дасть прогнозований клас. Матриця невідповідностей не показує емпіричного ризику, а відображає частоту помилок out-of-bag (OOB), яка на відміну від емпіричного ризику є об'єктивною оцінкою помилки для набору [29, с. 150].

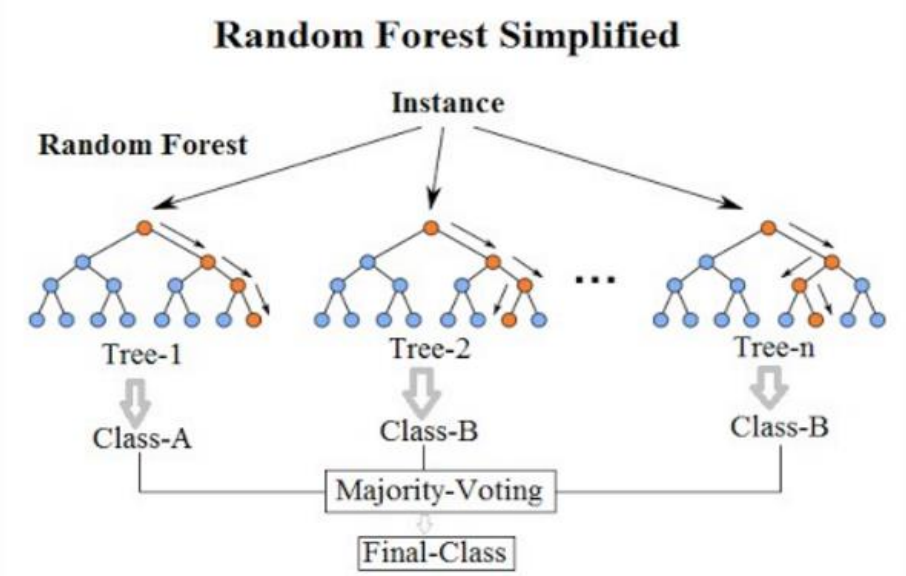


Рисунок 2.6 – Схема класифікатору випадкового лісу

Алгоритм випадкового лісу має низку ключових переваг і недоліків, коли він використовується для задач класифікації або регресії. Деякі з них включають [44]:

- Зменшення ризику перенавчання: дерева рішень мають ризик перенавчання, оскільки вони, як правило, щільно підходять для всіх зразків у навчальних даних. Однак, коли у випадковому лісі існує значна кількість дерев рішень, класифікатор не буде перенавчати модель, оскільки усереднення некорельованих дерев зменшує загальну дисперсію та помилку передбачення.

- Забезпечується гнучкість, оскільки випадковий ліс може обробляти задачі як регресії, так і класифікації з високим ступенем точності, це популярний метод серед дослідників даних. Бутстрепова агрегація також робить класифікатор

випадкового лісу ефективним інструментом для оцінки відсутніх значень, оскільки вона підтримує точність, коли частина даних відсутня.

- Легко визначити важливість функції: Випадковий ліс дозволяє легко оцінити важливість змінної або внесок у модель. Процес займає багато часу, оскільки алгоритми випадкового лісу можуть обробляти великі набори даних, вони можуть надавати точніші прогнози, але можуть повільно обробляти дані, оскільки вони обчислюють дані для кожного окремого дерева рішень. Вимагає більше ресурсів, оскільки випадковий ліс обробляє більші набори даних, їм знадобиться більше ресурсів для зберігання цих даних. Більш складний, оскільки прогноз окремого дерева рішень легше інтерпретувати порівняно з лісом із них.

2.3 Використання ансамблів моделей як більш ефективного алгоритму класифікації

Для методів ансамблевих моделей в машинному навчанні характерний зсув та/або дисперсія. Зсув – це різниця між прогнозованим значенням і фактичним значенням за моделлю. Зсув вводиться, коли модель не враховує варіації даних і створює просту модель. Проста модель не дотримується шаблонів даних, а, отже, модель дає помилки в прогнозуванні навчання, а також в даних тестування, тобто модель із високим зсувом і високою дисперсією [18].

Техніка ансамблювання (тобто комбінування різних моделей для створення однієї «оптимальної») дуже популярна у машинному навчанні. Це дає можливість не покладатися на одну єдину модель, яка може бути перенавчена або мати інші недоліки [45].

Тому для підвищення точності (оцінки) моделі розроблені методики ансамблевого навчання. Ансамбль — це концепція машинного навчання, у якій кілька моделей навчаються за допомогою алгоритмів машинного навчання. Він поєднує низькоефективні класифікатори (також звані слабкими учнями або базовими учнями) і комбінує індивідуальне прогнозування моделі для остаточного прогнозування [46, с. 101].

На основі типу базових учнів (базових моделей) ансамблеві методи можна класифікувати як однорідні та гетерогенні ансамблеві методи. Якщо базові учні однакові, то це метод однорідного ансамблю. Якщо базові учні різні, то це метод гетерогенного ансамблю [47, с. 279].

Ансамблеві техніки поділяються на три види:

- Bagging (беггінг);
- Boosting (бустинг);
- Stacking (стекінг).

Bagging –це абревіатура від «Bootstrap Aggregation» та використовується для зменшення дисперсії в моделі прогнозування. Bagging – це паралельний метод, який підходить для різних учнів, які вважаються незалежними один від одного, що дає змогу навчати їх одночасно [48, с. 171].

Bagging генерує додаткові дані для навчання з набору даних. Це досягається шляхом випадкової вибірки із заміною вихідного набору даних. Вибірка із заміною може повторювати деякі спостереження в кожному новому навчальному наборі даних. Кожен елемент у Bagging з однаковою ймовірністю з’явиться в новому наборі даних [48, с. 173].

Ці набори даних використовуються для паралельного навчання кількох моделей. Розраховується середнє значення всіх прогнозів з різних моделей ансамблю. Під час класифікації враховується більшість голосів, отриманих за допомогою механізму голосування [48, с. 173].

Модель Random Forest також використовує Bagging, де присутні моделі дерева рішень з більшою дисперсією. Він робить випадковий вибір функцій для вирощування дерев. Кілька випадкових дерев утворюють випадковий ліс [49].

Boosting — це послідовний метод ансамблю, який ітеративно регулює вагу спостереження відповідно до останньої класифікації. Якщо спостереження неправильно класифіковано, це збільшує вагу цього спостереження. Термін «бустинг» мовою непрофесіоналів стосується алгоритмів, які перетворюють

слабкого учня на сильнішого. Це зменшує помилку зсуву та створює надійні прогностичні моделі [50].

Точки даних, неправильно спрогнозовані під час кожної ітерації, виявляються, а їх ваги збільшуються. Алгоритм Boosting призначає ваги кожній отриманій моделі під час навчання. Учневі з хорошими результатами прогнозування даних навчання буде присвоєно вищу вагу. Під час оцінювання нового учня Boosting відстежує помилки учня [50].

Якщо наданий вхід є невідповідним, його вага збільшується. Мета, що стоїть за цим, полягає в тому, щоб майбутня гіпотеза з більшою ймовірністю належним чином класифікувала його шляхом об’єднання всього набору, щоб нарешті перетворити слабких учнів на моделі з кращою ефективністю [51, с. 32].

Він включає в себе кілька алгоритмів посилення. Оригінальні алгоритми не були адаптивними. Вони не могли максимально використати слабких учнів. Потім винайшли AdaBoost, який є адаптивним алгоритмом бустингу. Він отримав поважну премію Геделя та став першим успішним алгоритмом підвищення, створеним для двійкової класифікації. AdaBoost означає Adaptive Boosting. Він об’єднує кілька «слабких класифікаторів» в «сильний класифікатор» [52].

Gradient Boosting є розширенням процедури бустингу. Він використовує алгоритм градієнтного спуску, здатний оптимізувати будь-яку диференційовану функцію втрат. Його робота передбачає побудову ансамблю дерев, причому окремі дерева підсумовуються послідовно. Наступне дерево відновлює втрату (різниця між реальними та прогнозованими значеннями) [53].

Stacking є одним із популярних методів моделювання ансамблю в машинному навчанні. Різні слабкі учні об’єднуються паралельно таким чином, щоб, об’єднавши їх із метаучнями, можливо було передбачити кращі прогнози на майбутнє [54, с. 163].

Ця методика ансамблю працює шляхом застосування комбінованих прогнозів кількох слабких учнів і метанавчань, щоб можна було досягти кращої моделі прогнозування результатів [54, с. 163].

У стекуванні алгоритм приймає вихідні дані підмоделей як вхідні дані та намагається дізнатися, як найкраще об’єднати вхідні прогнози, щоб зробити кращий вихідний прогноз [52].

Стекінг також відомий як узагальнення з нагромадженням і є розширеною формою методики ансамблю усереднення моделі, в якій усі підмоделі однаково беруть участь відповідно до їх ваг продуктивності та створюють нову модель із кращими прогнозами. Ця нова модель розташована поверх інших, це причина, чому його назвали стекуванням [51, с. 22].

Архітектура стекової моделі розроблена таким чином, що вона складається з двох або більше базових моделей/моделей учня та метамоделі, яка поєднує прогнози базових моделей. Ці базові моделі називають моделями рівня 0, а метамодель — моделлю рівня 1. Таким чином, метод сумісного ансамблю включає вихідні (навчальні) дані, моделі первинного рівня, прогноз первинного рівня, модель вторинного рівня та остаточний прогноз [54].

2.4 Метрики оцінки якості роботи класифікаторів

Досі точність класифікатора вимірювалася як кількість правильних прогнозів, поділена на загальну кількість прогнозів. Таким чином отримували частку випадків, у яких алгоритм навчання мав рацію. Однак було б розумно зібрати додаткову інформацію щодо ефективності класифікатора, перш ніж використовувати його на практиці для прогнозування. Оцінюючи класифікатора необхідно пам'ятати про проблему дисбалансу класів, що виникає, коли переважна більшість записів у наборі даних відносяться до одного класу. Ця проблема є і в досліджуваному наборі даних [55].

Є багато способів виміряти ефективність класифікатора, проте найкращий показник – той, який визначає, чи є класифікатор успішним, коли застосовується за прямим призначенням. Важливо визначити показники ефективності, які показуватимуть корисність, а чи не просто точність. Альтернативні показники ефективності отримаємо із матриці невідповідностей [55].

Якщо прогнозоване значення збігається з реальним, класифікація правильна. Правильні прогнози розміщуються у матриці невідповідностей щодо діагоналі. Інші комірки матриці відповідають випадкам, коли прогнозоване значення відрізняється від реального. Це неправильні прогнози. Показники ефективності для моделей класифікації засновані на кількості прогнозів, що потрапляють на діагональ та за її межі [25, с. 671].

Найбільш поширені показники ефективності враховують здатність моделі відрізняти один клас від інших. У цьому цікавий клас називається позитивним, інші – негативним [25, с. 672].

Залежність між прогнозами позитивного та негативного класу може бути представлена у вигляді матриці невідповідностей 2x2, у якій показано, чи відносяться прогнози до однієї з чотирьох категорій [56]:

- Істинно-позитивна (True Positive, TP) - правильно класифікується як клас, що цікавить;

- Істинно-негативна (True Negative, TN) – правильно класифікований як той, що не належить до класу, що цікавить;

- Хибно-позитивна (False Positive, FP) – неправильно класифікується як клас, що цікавить;

- Хибно-негативна (False Negative, FN) – неправильно класифікована як така, що не належить до цікавого класу.

Подібна матриця невідповідностей є основою багатьох найважливіших показників ефективності моделі [56].

За допомогою матриці невідповідностей 2x2 можна формалізувати визначення точності прогнозування (accuracy) (іноді називається *коефіцієнтом успішності*) [57]:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (2.7)$$

У цій формулі TP, TN, FP та FN позначають кількість випадків, коли прогнози моделі потрапляють у кожну з відповідних категорій. Таким чином,

точність є відношенням суми істинно позитивних і істинно негативних значень до загальної кількості прогнозів [57].

Коефіцієнт помилок (error rate), або частка невірно класифікованих прикладів визначається так [57]:

$$\text{error rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = 1 - \text{accuracy} \qquad (2.8)$$

Значення коефіцієнта помилок можна обчислити як 1 мінус точність. Наприклад, якщо модель є вірною у 95% випадків, то у 5% випадків вона неправильна [57].

Каппа-статистика коригує значення точності з огляду на те, що правильний прогноз може бути виданий випадково. Це особливо важливо для наборів даних із серйозним дисбалансом класів, оскільки класифікатор може отримати високе значення точності лише тому, що випадково вгадує найчастіший клас. Високе значення каппа-статистики класифікатор може набути тільки в тому випадку, якщо дає вірні прогнози частіше, ніж за такою спрощеною стратегією [58].

Значення каппа-статистики лежать у діапазоні від 0 до 1. Одиниця вказує на ідеальну відповідність прогнозів моделі та справжніх значень. Значення менше одиниці вказують на неідеальну відповідність. Залежно від того, як використовуватиметься модель, можливі різні варіанти інтерпретації каппа-статистики. Ось один із типових варіантів [58]:

- Погана відповідність – менше 0,20;
- Задовільна відповідність – від 0,20 до 0,40;
- Середня відповідність – від 0,40 до 0,60;
- Хороша відповідність – від 0,60 до 0,80;
- Дуже гарна відповідність – від 0,80 до 1,00.

Важливо пам'ятати, що ці категорії суб'єктивні. Нижче наведено формулу для обчислення каппа-статистики. У цій формулі Pr (a) означає пропорцію реальної відповідності, а Pr (e) – очікувану відповідність між прогнозом класифікатора та істинними значеннями за умови, що вони обрані випадковим чином [59, с. 125]:

$$Kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \quad (2.9)$$

Підібрати корисний класифікатор часто означає знайти баланс між надмірно консервативними та надмірно агресивними прогнозами. Цей компроміс відбиває наступна пара показників ефективності: чутливість та специфічність [60, с. 121].

Чутливість (sensitivity) моделі (звана частотою істинно позитивних прогнозів) – це частка правильно класифікованих позитивних прикладів. Тому, чутливість обчислюється як кількість істинно позитивних прогнозів, поділена на загальну кількість позитивних результатів, класифікованих як правильно (істинно-позитивні), так і помилково (хибно-негативних) [60, с. 122].

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.10)$$

Специфічність (specificity) моделі (чи частота істинно-негативних прогнозів) – це частка правильно класифікованих негативних прикладів. Подібно до чутливості, специфічність обчислюється як число істинно негативних прогнозів, розділене на загальну кількість негативних результатів – як істинно негативних, так і хибно-позитивних [60, с. 122].

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad (2.11)$$

Точність (або прогностична цінність позитивних результатів) визначається як частка реальних позитивних прикладів, які прогнозовані як позитивні. Інакше кажучи, як часто модель правильно прогнозує позитивний клас? Точна модель прогнозуватиме позитивний клас лише у тих випадках, які справді можуть бути позитивними. У разі фільтру не банкрутів висока точність означає, що модель здатна точно фільтрувати тільки не банкрутів, пропускаючи банкрутів [61, с. 245].

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.12)$$

Повнота (recall), навпаки, є показником того, наскільки повними є результати. Повнота окреслюється частка істинно позитивних прогнозів у кількості позитивних прогнозів [61, с. 246].

Модель з високим рівнем повноти фіксує більшу частину позитивних прикладів, а це означає, що вона має широке охоплення. Наприклад, пошукова система з високою повнотою повертає велику кількість документів, що відповідають пошуковому запиту. Аналогічно фільтр не банкрут має високий рівень повноти, якщо правильно ідентифікує більшість не банкрутів [55].

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad (2.13)$$

Міра ефективності моделі, яка поєднує точність і повноту в одне число називається *F*-мірою (F-measure) (також званою F1 або F-оцінкою). F-мера поєднує у собі точність і повноту, використовуючи середнє гармонійне – тип середнього, який визначає швидкість виміру величини. F-міра обчислюється за формулою [56]:

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \qquad (2.14)$$

ROC крива (receiver operating characteristic curve) - це граф, що показує продуктивність моделі класифікації за всіх можливих значеннях класифікаційних порогів. Ця крива – графік двох параметрів [56]:

- Істинно позитивна швидкість (True Positive Rate)
- Помилково позитивна швидкість (False Positive Rate)

Істинно позитивна швидкість (TPR) - це синонім відгуку і тому визначається таким чином [62, с. 275]:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad (2.15)$$

Помилково позитивна швидкість (FPR) визначається наступним чином [56]:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{FN}} \qquad (2.16)$$

Крива ROC є графіком TPR та FPR при різних значеннях порогів класифікації. Зниження порогу класифікації класифікує більше прикладів як позитивні, при цьому збільшуються і помилково позитивні та істинно позитивні. Наступний графік показує типову криву ROC [61, с. 253].

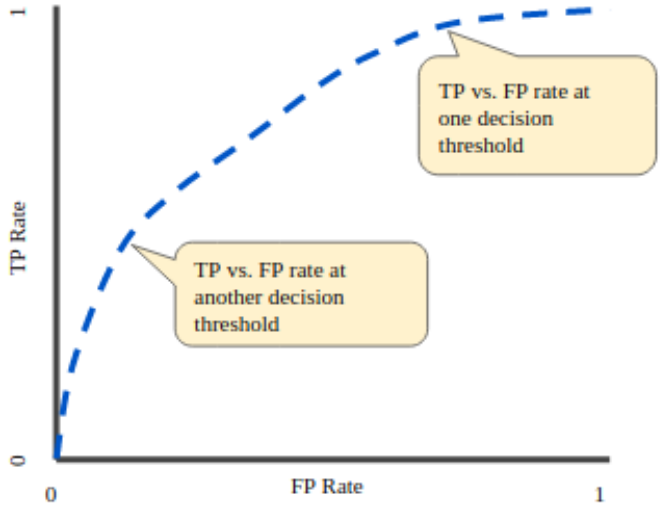


Рисунок 2.6 – TP та FP швидкість при різних класифікаційних порогах

Щоб обчислити точки на кривій ROC, ми можемо багато разів оцінювати модель регресії логістики з різними порогами класифікації, але це буде неефективним. На щастя, існує ефективний, заснований на сортуванні, алгоритм, який може надати нам цю інформацію під назвою AUC [58].

AUC позначає «область під кривою ROC – Area under the ROC Curve». AUC вимірює всю двовимірну область під весь ROC-кривою, тобто обчислює інтеграл від (0,0) до (1,1). AUC надає сукупний вимір продуктивності за всіх можливих значень класифікаційного порога. Один із шляхів інтерпретувати AUC - розглядати її як ймовірність, що модель ранжує випадковий позитивний приклад вище, ніж випадковий негативний приклад [57].

Висновки до розділу 2

1. Попередня обробка даних – це процес отримання необроблених даних та перетворення їх у чисті, сформовані набори, які дозволяють проводити інтелектуальний аналіз, обробку та аналіз даних.

2. Класифікація — це двоетапний процес у машинному навчанні, етап навчання та етап прогнозування.

3. Попередня обробка даних може допомогти досягти кращих результатів.

4. Необроблені дані зазвичай мають наступні характеристики: дублювання, невідповідності, шуми, викиди, порожні або NaN значення, аномалії та безліч інших проблем.

5. П’ять кроків для попередньої обробки даних: оцінка даних, очищення даних, інтеграція та перетворення даних, скорочення та зразок даних.

6. У машинному навчанні класифікація відноситься до задачі прогнозного моделювання, коли мітка класу прогнозується для вхідних даних.

7. У якості базових моделей обрано моделі: дерева рішень, Наївний Баєсів класифікатор, лінійний дискримінантний аналіз, квадратичний дискримінантний аналіз, логістична регресія, метод опорних векторів, метод найближчого сусіда, штучні нейронні мережі та модель випадкового лісу.

8. Ансамбль — це концепція машинного навчання, у якій кілька моделей навчаються за допомогою алгоритмів машинного навчання.

9. Поєднуються низькоефективні класифікатори (також звані слабкими учнями або базовими класифікаторами) і комбінується індивідуальне прогнозування моделі для остаточного прогнозування.

10. Є багато способів виміряти ефективність класифікатора, проте найкращий показник – той, який визначає, чи є класифікатор успішним, коли застосовується за прямим призначенням.

11. Найбільш поширені показники ефективності враховують здатність моделі відрізняти один клас від інших: коефіцієнти успішності та помилок, каппа-статистика, чутливість, точність, повнота, F-міра та область під кривою ROC.

3 ЗБІР ДАНИХ, АНАЛІЗ ТА ЇХ ІНТЕРПРЕТАЦІЯ. ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

3.1 Загальний опис набору даних

Для створення інтелектуальної системи класифікації на основі ансамблів моделей було використано набір даних E-Commerce Shipping Data. Мета моделі – виявити чинники, пов'язані з ризиком невчасної доставки товару клієнту. Для цього необхідно отримати дані про велику кількість попередніх замовлень клієнтами та інформацію про одержувачів товарів.

Дані з такими характеристиками доступні в наборі даних, переданих Prachi Gopalani з Джодхпурського інституту в репозиторій машинного навчання Kaggle. Цей набір даних містить інформацію про дані доставки, отримані від міжнародної E-Commerce компанії Індії [63].

Проаналізуємо набір даних, наданий міжнародною компанією, яка хоче отримати дані сервісу доставки зі своєї бази даних клієнтів. Набір даних, використаний для побудови моделі, містить 10999 спостережень та 12 змінних. Дані містять таку інформацію як:

- ідентифікаційний номер клієнтів,
- складський блок (компанія має великий склад, який розділений на блоки, такі як A, B, C, D, E),
- спосіб доставки (компанія доставляє продукцію кількома способами, наприклад кораблем, літаком і наземними транспортними засобами),
- дзвінки у службу підтримки клієнтів (кількість дзвінків, здійснених клієнтами щодо запиту про відправлення товару),
- рейтинг клієнтів (компанія оцінила кожного клієнта: 1 — найнижча (найгірша), 5 — найвища (найкраща) оцінки),
- вартість продукту (вартість продукту в доларах США),
- попередні покупки (кількість попередніх покупок),

- важливість продукту (компанія класифікувала товари за різними ступенями важливості, такими як: низький, середній, високий),
- стать покупця (чоловіча та жіноча),
- знижка (пропонується на певний продукт),
- вага в грамах,
- доставка товару (значення «1» вказує, що продукт не доставлено вчасно, а «0» вказує на те, що він доставлений вчасно).

Набір даних включає 10999 прикладів доставки товару, а також набір числових і номінальних ознак, що визначають характеристики товару та клієнта. Змінна *Reached.on.Time_Y.N* є результатом класифікації і вказує на те, чи доставлений вчасно товар чи ні.

Для вирішення задачі класифікації використовуються різноманітні методи і моделі машинного навчання, для реалізації програмних модулів використовується мова програмування R, та середовище розробки RStudio. Методи об’єднані в єдину методологію класифікації.

3.2 Аналіз та опис структури набору даних

Імпортування даних виконується за допомогою функції *read.table()*. Попередньо необхідно звернути увагу на розташування вихідного набору даних у поточному каталозі, вказавши його посилання:

```
> setwd('D:\\')
> dat <- read.csv("Shipping.csv")
```

В результаті завантаження створюється кадр даних про сервіс доставки з декількома факторними змінними.

За допомогою функції *names()* наведено відповідну таблицю назв змінних для цього набору даних.

```
> names(dat)
 [1] "ID"                 "Warehouse_block"    "Mode_of_Shipment"
 [4] "Customer_care_calls" "Customer_rating"    "Cost_of_the_Product"
 [7] "Prior_purchases"    "Product_importance" "Gender"
[10] "Discount_offered"   "Weight_in_gms"      "Reached.on.Time_Y.N"
```

Виведення структури набору даних для подальшого аналізу виконується за допомогою функції *str()*:

```
> str(dat)
'data.frame':   10999 obs. of  12 variables:
 $ ID                 : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Warehouse_block    : chr  "D" "F" "A" "B" ...
 $ Mode_of_Shipment   : chr  "Flight" "Flight" "Flight" "Flight" ...
 $ Customer_care_calls: int  4 4 2 3 2 3 3 4 3 3 ...
 $ Customer_rating    : int  2 5 2 3 2 1 4 1 4 2 ...
 $ Cost_of_the_Product: int  177 216 183 176 184 162 250 233 150 164 ...
 $ Prior_purchases    : int  3 2 4 4 3 3 3 2 3 3 ...
 $ Product_importance : chr  "low" "low" "low" "medium" ...
 $ Gender             : chr  "F" "M" "M" "M" ...
 $ Discount_offered   : int  44 59 48 10 46 12 3 48 11 29 ...
 $ Weight_in_gms      : int  1233 3088 3374 1177 2484 1417 2371 2804 1861 1187 ...
 $ Reached.on.Time_Y.N: int  1 1 1 1 1 1 1 1 1 1 ...
```

До нечислових або факторних змінних відносяться атрибути: *Warehouse_block*, *Mode_of_Shipment*, *Product_Importance* та *Gender*. Змінні *ID*, *Customer_care_calls*, *Customer_rating*, *Cost_of_the_Product*, *Prior_purchases*, *Discount_offered* та *Weight_in_gms* дійсно кількісні, а також результат класифікації атрибут *Reached.on.Time_Y.N*.

Переглянемо набір даних за допомогою команди *head()* та *summary()*. У даному випадку *head* виводить дані про перші 6 замовлень, а *summary* – виводить узагальнену інформацію про *dat*, це набір статистичних параметрів, що описують: Min: мінімальне значення, 1st Qu: значення 1-го квартиля (25-й процентиль), Median: медіанне значення, Mean: середнє арифметичне, 3-я Qu: значення 3-го квартиля (75-й процентиль), Max: максимальне значення.

```
> head(dat, n = 6)
  ID Warehouse_block Mode_of_Shipment Customer_care_calls Customer_rating
1  1               D           Flight                   4               2
2  2               F           Flight                   4               5
3  3               A           Flight                   2               2
4  4               B           Flight                   3               3
5  5               C           Flight                   2               2
6  6               F           Flight                   3               1
  Cost_of_the_Product Prior_purchases Product_importance Gender Discount_offered
1                 177               3                low      F               44
2                 216               2                low      M               59
3                 183               4                low      M               48
4                 176               4             medium      M               10
5                 184               3             medium      F               46
6                 162               3             medium      F               12
  Weight_in_gms Reached.on.Time_Y.N
1          1233                   1
2          3088                   1
3          3374                   1
4          1177                   1
5          2484                   1
6          1417                   1
```

Деякі стовпці мають неправильну структуру, тож перш ніж виконувати команду *summary*, виконаємо зміну типів даних. Дані в стовпцях *Warehouse_block, Mode_of_Shipment,Product_Importance,Reached.on.Time_Y.N* є категоріальними даними. Отже, ми повинні змінити структуру цих стовпців до типу даних Factor.

```
> summary(dat)
       ID        Warehouse_block Mode_of_Shipment Customer_care_calls
 Min.   :    1   A:1833          Flight:1777      Min.   :2.000
 1st Qu.: 2750   B:1833          Road  :1760      1st Qu.:3.000
 Median : 5500   C:1833          Ship  :7462      Median :4.000
 Mean   : 5500   D:1834                           Mean   :4.054
 3rd Qu.: 8250   F:3666                           3rd Qu.:5.000
 Max.   :10999                                    Max.   :7.000
 Customer_rating Cost_of_the_Product Prior_purchases  Product_importance Gender
 Min.   :1.000   Min.   : 96.0       Min.   : 2.000   high  : 948        F:5545
 1st Qu.:2.000   1st Qu.:169.0       1st Qu.: 3.000   low   :5297        M:5454
 Median :3.000   Median :214.0       Median : 3.000   medium:4754
 Mean   :2.991   Mean   :210.2       Mean   : 3.568
 3rd Qu.:4.000   3rd Qu.:251.0       3rd Qu.: 4.000
 Max.   :5.000   Max.   :310.0       Max.   :10.000
 Discount_offered Weight_in_gms  Reached.on.Time_Y.N
 Min.   : 1.00    Min.   :1001   0:4436
 1st Qu.: 4.00    1st Qu.:1840   1:6563
 Median : 7.00    Median :4149
 Mean   :13.37    Mean   :3634
 3rd Qu.:10.00    3rd Qu.:5050
 Max.   :65.00    Max.   :7846
```

Проаналізувавши отримані результати за допомогою *summary*, можна дійти наступних висновків:

a) більшість товарів відправляється зі складського блоку F;

b) більшість товарів доставляється кораблем;

c) в середньому клієнти здійснюють 4 дзвінки у службу підтримки;

d) середня оцінка рейтингу клієнта: 2.991;

e) більшість товарів мають низький і високий пріоритет;

f) клієнтів-жінок трохи більше, ніж клієнтів-чоловіків;

g) більшість знижок менше 10%;

h) середня вага товару становить 3,36 кг.;

i) більшість товарів доставляється невчасно.

Провівши додатковий аналіз, видаляються ознаки, що не мають безпосереднього впливу на результуючу змінну *Reached.on.Time_Y.N*. В нашому

випадку – це єдина зміна *ID*. В результаті відбору ознак набір даних матиме таку структуру:

```
'data.frame':   10999 obs. of  11 variables:
 $ Warehouse_block    : Factor w/ 5 levels "A","B","C","D",..: 4 5 1 2 3 5 4 5 1 2$
 $ Mode_of_Shipment   : Factor w/ 3 levels "Flight","Road",..: 1 1 1 1 1 1 1 1 1 1$
 $ Customer_care_calls: num  4 4 2 3 2 3 3 4 3 3 ...
 $ Customer_rating    : num  2 5 2 3 2 1 4 1 4 2 ...
 $ Cost_of_the_Product: num  177 216 183 176 184 162 250 233 150 164 ...
 $ Prior_purchases    : num  3 2 4 4 3 3 3 2 3 3 ...
 $ Product_importance : Factor w/ 3 levels "high","low","medium": 2 2 2 3 3 3 2 2 $
 $ Gender             : Factor w/ 2 levels "F","M": 1 2 2 2 1 1 1 1 1 1 ...
 $ Discount_offered   : num  44 59 48 10 46 12 3 48 11 29 ...
 $ Weight_in_gms      : num  1233 3088 3374 1177 2484 ...
 $ Reached.on.Time_Y.N: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Розглянемо деякі показники набору даних у відношенні до результуючої змінної у графічному вигляді.

- Відсотковий розподіл доставленого товару.

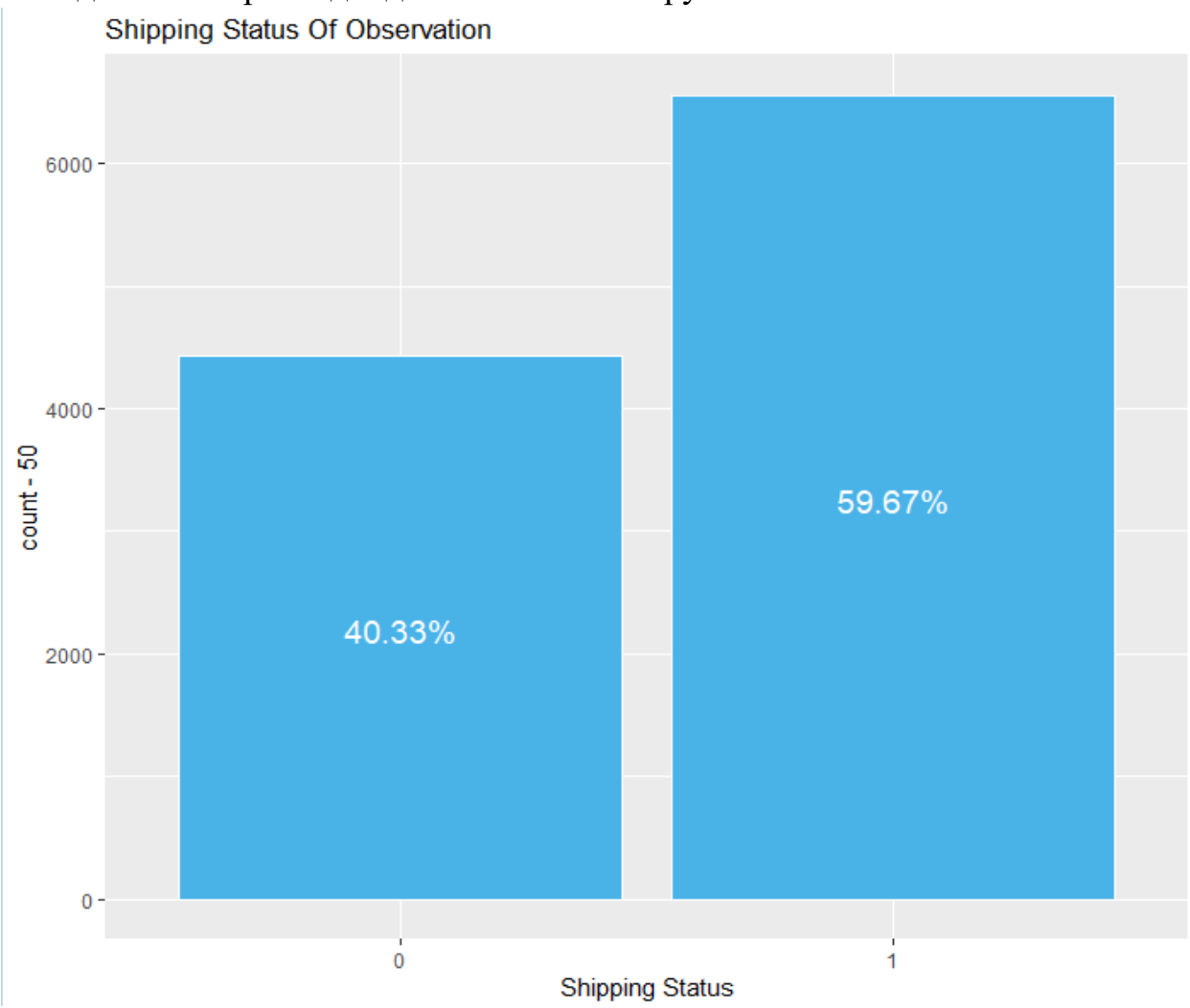


Рисунок 3.1 – Відсотковий розподіл по результуючій змінній

На рисунку 3.1 результатів видно, що кількість невчасно доставлених відправлень (59,67%) перевищує доставку у вказані терміни (40,33%).

- Співвідношення типу складського блоку до статусу доставки.

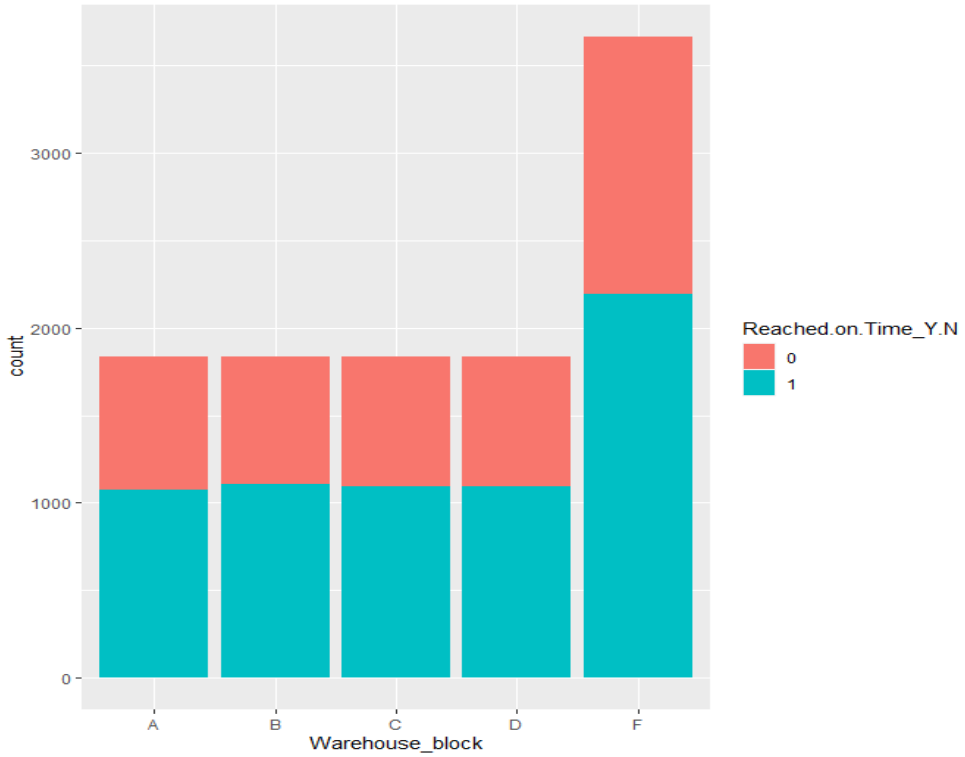


Рисунок 3.2 – Відношення змінної *Warehouse_Block* до результуючої

З наведеного результату на рисунку 3.2, базуючись на змінній *Warehouse_block*, видно, що кількість доставок невчасно більше, ніж вчасно.

- Співвідношення типу доставки до терміну доставки.

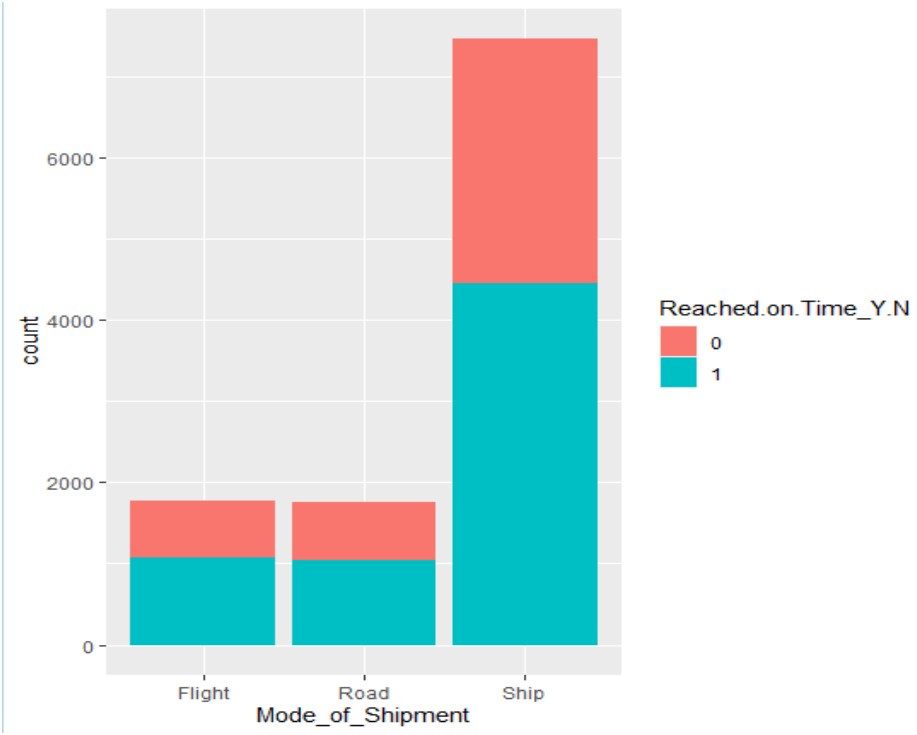


Рисунок 3.3 – Відношення змінної *Mode_of_Shipment* до результуючої

У відношенні до способу відправлення, на рисунку 3.3 видно, що кількість невчасно доставленого товару є більшою, ніж кількість вчасного.

- Співвідношення рейтингу клієнта на основі статусу доставки.

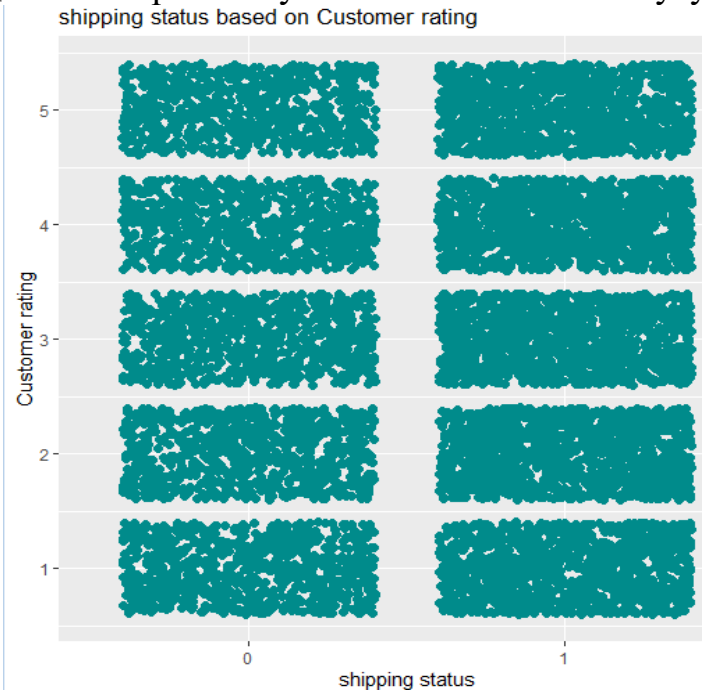


Рисунок 3.4 – Відношення змінної *Customer_rating* до результуючої

На рисунку 3.4 видно, що на основі рейтингової оцінки клієнтів компанією, кількість невчасно перевищує кількість вчасно доставлених товарів.

- Співвідношення вартості продукту до статусу доставки.

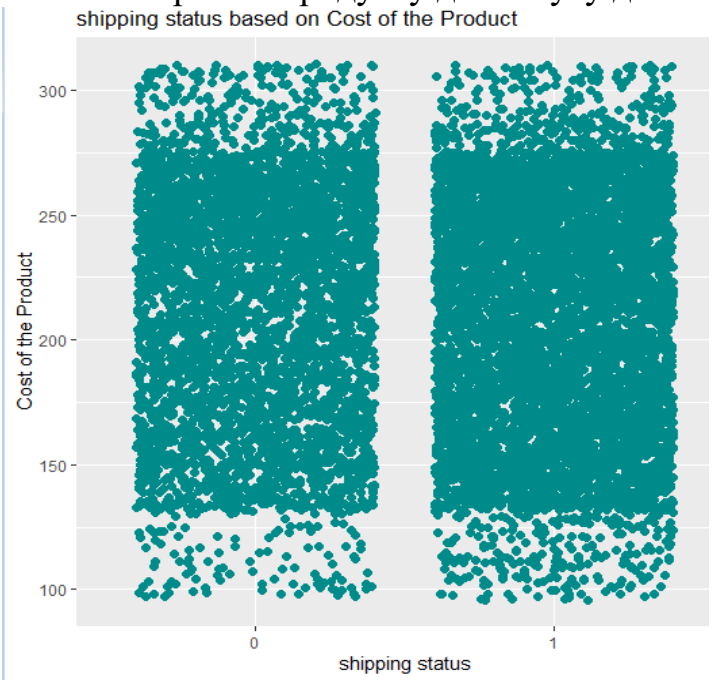


Рисунок 3.5 – Відношення змінної *Cost_of_the_Product* до результуючої

З наведеного вище рисунку ми бачимо, незалежно від вартості продукту, частка невчасно доставленого товару є більшою, ніж вчасно.

- Співвідношення важливості продукту до статусу його доставки.

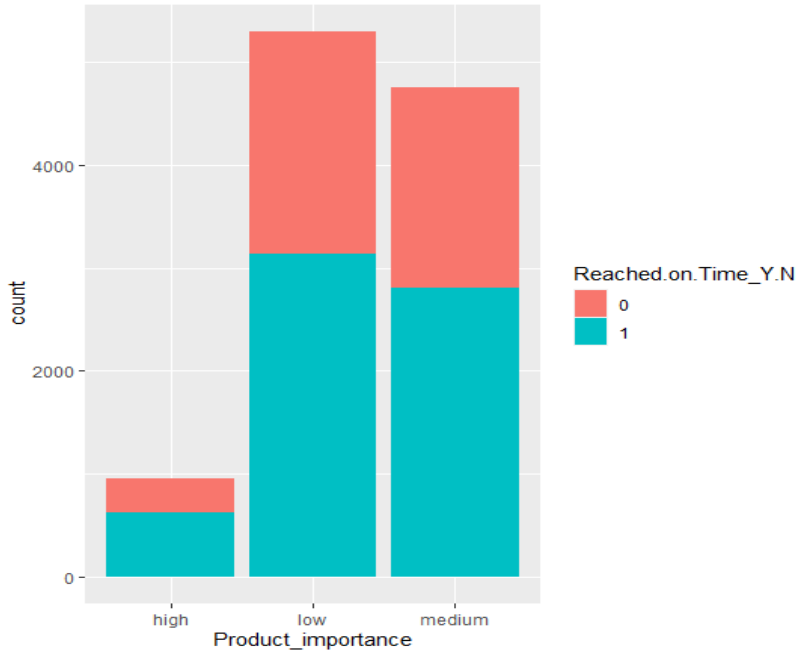


Рисунок 3.6 – Відношення змінної *Product_Importance* до результуючої

З рисунку 3.6 видно, що незалежно від важливості товару, кількість невчасної доставки є більшою, ніж вчасної. Причому, при високій важливості товару рівень запізнення з доставкою найбільший.

Проаналізувавши співвідношення деяких показників набору даних до результуючої змінної, можна стверджувати, що компанія має великі проблеми із вчасною доставкою товару. Але чому? Проаналізуємо та знайдемо головні причини затримок доставки.

```
> round(prop.table(table(dat$Reached.on.Time_Y.N, dat$Mode_of_Shipment), margin = 2)*100, 2)

    Flight  Road  Ship
  0  39.84 41.19 40.24
  1  60.16 58.81 59.76
> round(prop.table(table(dat$Reached.on.Time_Y.N, dat$Warehouse_block), margin = 2)*100, 2)

       A     B     C     D     F
  0 41.35 39.77 40.32 40.24 40.15
  1 58.65 60.23 59.68 59.76 59.85
> round(prop.table(table(dat$Reached.on.Time_Y.N, dat$Prior_purchases), margin = 2)*100, 2)

       2     3     4     5     6     7     8    10
  0 37.48 35.93 45.66 50.12 44.03 32.35 35.16 42.70
  1 62.52 64.07 54.34 49.88 55.97 67.65 64.84 57.30
> round(prop.table(table(dat$Reached.on.Time_Y.N, dat$Product_importance), margin = 2)*100, 2)

    high   low medium
  0 35.02 40.72  40.95
  1 64.98 59.28  59.05
```

Як ми бачимо, затримка відправлення для кожного виду транспорту, складського блоку, попередніх купівель та важливості продукту майже однакова, що становить близько 55% - 65%. Таким чином, ми можемо зробити висновок, що значення результуючої змінної «невчасно» не є єдиним для співвідношень з обраними змінними, а є й своєчасні відправлення. Створимо фрейм даних із комбінації наведених вище змінних.

```
> head(reach.time)
    Moda Prior Warehouse On.Time Late.Time Perc.OT Perc.LT Avg.Cost.OT Avg.Cost.LT Avg.Disc.OT
1 Flight  high         A      13        18   41.94   58.06      187.46      210.67    6.615385
2   Road  high         A       5        22   18.52   81.48      237.00      197.45    6.400000
3   Ship  high         A      36        71   33.64   66.36      206.78      204.46    5.361111
4 Flight   low         A      64        80   44.44   55.56      214.34      205.11    5.906250
5   Road   low         A      66        79   45.52   54.48      214.73      204.08    5.272727
6   Ship   low         A     242       378   39.03   60.97      216.98      205.77    5.500000
  Avg.Disc.LT Avg.Call.OT Avg.Call.LT
1    19.88889    3.923077    3.944444
2    18.45455    3.200000    3.636364
3    22.43662    4.083333    3.845070
4    16.63750    4.171875    4.112500
5    19.69620    4.060606    3.759494
6    19.10317    4.210744    3.949735
```

Визначимо найбільшу кількість комбінацій, що мають найвищий затримку доставки.

Нижче наведено 19 комбінацій видів транспортування, важливості продукту та складського блоку, що мають найбільшу затримку доставки. Топ-комбінація, яка призводить до найбільшої затримки доставки – це товари високої важливості, які доставляються кораблем. Важливі товари рекомендовано відправляти літаком.

Товари, доставлені із запізненням, коштують дешевше, ніж товари, які доставлені вчасно. Також товари, на які знижки більше 10%, доставляються із запізненням.

```
> most.late <- reach.time[reach.time$Perc.LT > mean(reach.time$Perc.LT), ]
> most.late[order(most.late$Prior, decreasing = F), ]
    Moda  Prior Warehouse On.Time Late.Time Perc.OT Perc.LT Avg.Cost.OT Avg.Cost.LT Avg.Disc.OT
2   Road   high         A       5        22   18.52   81.48      237.00      197.45    6.400000
3   Ship   high         A      36        71   33.64   66.36      206.78      204.46    5.361111
11  Road   high         B       9        20   31.03   68.97      207.78      190.35    5.111111
12  Ship   high         B      26        66   28.26   71.74      210.23      192.53    4.961538
19 Flight  high         C      14        22   38.89   61.11      183.86      195.23    4.000000
20  Road   high         C       8        19   29.63   70.37      195.12      182.32    5.625000
21  Ship   high         C      37        68   35.24   64.76      216.46      198.47    5.081081
30  Ship   high         D      42        75   35.90   64.10      212.12      208.83    5.523810
39  Ship   high         F      68       138   33.01   66.99      205.62      202.74    5.058824
6   Ship    low         A     242       378   39.03   60.97      216.98      205.77    5.500000
```

```
> summary(most.late)
     Moda        Prior     Warehouse    On.Time          Late.Time         Perc.OT          Perc.LT
 Flight:6   high   :9   A:4        Min.   :  5.00   Min.   : 19.0   Min.   :18.52   Min.   :60.97
 Road  :6   low    :5   B:5        1st Qu.: 31.00   1st Qu.: 67.0   1st Qu.:33.33   1st Qu.:62.16
 Ship  :7   medium:5   C:4        Median : 47.00   Median : 80.0   Median :36.84   Median :63.16
                       D:3        Mean   : 64.11   Mean   :109.7   Mean   :34.73   Mean   :65.27
                       F:3        3rd Qu.: 61.50   3rd Qu.:114.5   3rd Qu.:37.84   3rd Qu.:66.67
                                  Max.   :242.00   Max.   :378.0   Max.   :39.03   Max.   :81.48
  Avg.Cost.OT     Avg.Cost.LT     Avg.Disc.OT     Avg.Disc.LT     Avg.Call.OT     Avg.Call.LT
 Min.   :183.9   Min.   :182.3   Min.   :4.000   Min.   :15.35   Min.   :3.200   Min.   :3.636
 1st Qu.:208.2   1st Qu.:198.0   1st Qu.:5.102   1st Qu.:17.74   1st Qu.:3.913   1st Qu.:3.854
 Median :212.1   Median :205.9   Median :5.400   Median :18.38   Median :4.091   Median :3.950
 Mean   :212.0   Mean   :203.1   Mean   :5.378   Mean   :18.49   Mean   :3.995   Mean   :3.929
 3rd Qu.:216.7   3rd Qu.:208.4   3rd Qu.:5.629   3rd Qu.:19.06   3rd Qu.:4.226   3rd Qu.:4.006
 Max.   :237.0   Max.   :213.6   Max.   :6.400   Max.   :22.44   Max.   :4.438   Max.   :4.284
```

Товари, які доставляються вчасно, мають більшу кількість дзвінків, здійснених клієнтами щодо запиту про відправлення придбаного товару, це означає, що співробітники працюватимуть, якщо клієнти їм телефонують.

```
> aggregate(dat$Customer_rating ~ dat$Reached.on.Time_Y.N, FUN = mean)
  dat$Reached.on.Time_Y.N dat$Customer_rating
1                       0            2.967989
2                       1            3.005790
```

Але товари, доставлені із запізненням, мають трохи кращий рейтинг, ніж товари, доставлені вчасно. Це вказує на те, що час доставки не є показником від покупця для визначення рейтингу.

3.3 Попередня обробка даних. Підготовка до моделювання

Попередня обробка даних включає в себе перетворення категоріальних даних в числові, нормалізацію даних, перевірку на нечислові та відсутні значення, перевірку на аномальні значення та відбір ознак.

Першим кроком перетворимо факторних змінні *Warehouse_block (A→1, B→2, C→3, D→4, E→5)*, *Mode_of_Shipment (Flight→1, Ship→2, Road→3)*, *Product_Importance (high→1, medium→2, low→3)* та *Gender (F→1, M→2)* на числові, оскільки для більшості моделей машинного навчання на вхід подаються числові значення, а також для стандартизації значень атрибутів з метою отримання кращої точності навчання.

```
> dat$Gender<-as.numeric(ifelse(dat$Gender=='F',1,2))
> dat$Product_importance<-as.numeric(ifelse(dat$Product_importance=='high',1, ifelse
> dat$Mode_of_Shipment<-as.numeric(ifelse(dat$Mode_of_Shipment=='Flight',1, ifelse(d
> dat$Warehouse_block<-as.numeric(ifelse(dat$Warehouse_block=='A',1, ifelse(dat$Ware
> str(dat)
'data.frame':   10999 obs. of  11 variables:
 $ Warehouse_block    : num  4 5 1 2 3 5 4 5 1 2 ...
 $ Mode_of_Shipment   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Customer_care_calls: num  4 4 2 3 2 3 3 4 3 3 ...
 $ Customer_rating    : num  2 5 2 3 2 1 4 1 4 2 ...
 $ Cost_of_the_Product: num  177 216 183 176 184 162 250 233 150 164 ...
 $ Prior_purchases    : num  3 2 4 4 3 3 3 2 3 3 ...
 $ Product_importance : num  3 3 3 2 2 2 3 3 3 2 ...
 $ Gender             : num  1 2 2 2 1 1 1 1 1 1 ...
 $ Discount_offered   : num  44 59 48 10 46 12 3 48 11 29 ...
 $ Weight_in_gms      : num  1233 3088 3374 1177 2484 ...
 $ Reached.on.Time_Y.N: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Наступним кроком виконаємо нормалізацію даних за допомогою мінімаксного методу. Як бачимо змінні *Cost_of_the_Product*, *Discount_offered* та *Weight_in_gms* мають достатньо великі значення та всі дані з набору необхідно привести до загальної шкали без втрати інформації про відмінність діапазонів.

```
> normalize <- function(x) {
+ return ((x - min(x)) / (max(x) - min(x))) }
> dat <- as.data.frame(lapply(dat, normalize))
> fix(dat)
> head(dat, n = 6)
  Warehouse_block Mode_of_Shipment Customer_care_calls Customer_rating Cost_of_the_Product Prior_purchases
1            0.75                0                0.4           0.25           0.3785047           0.125
2            1.00                0                0.4           1.00           0.5607477           0.000
3            0.00                0                0.0           0.25           0.4065421           0.250
4            0.25                0                0.2           0.50           0.3738318           0.250
5            0.50                0                0.0           0.25           0.4112150           0.125
6            1.00                0                0.2           0.00           0.3084112           0.125
  Product_importance Gender Discount_offered Weight_in_gms Reached.on.Time_Y.N
1                1.0      0         0.671875    0.03389335                  1
2                1.0      1         0.906250    0.30489408                  1
3                1.0      1         0.734375    0.34667641                  1
4                0.5      1         0.140625    0.02571220                  1
5                0.5      0         0.703125    0.21665449                  1
6                0.5      0         0.171875    0.06077429                  1
```

Нечислові та відсутні значення в наборі даних перевіряються за допомогою функцій *NaValue* та *BlankValue*.

При перевірці з’ясовано, що в обраному для досліджень наборі даних немає нечислових чи відсутніх значень в жодному атрибуті.

```
> NaValue = function (x) {sum(is.na(x)) }
> apply(dat, 2, NaValue)
    Warehouse_block    Mode_of_Shipment Customer_care_calls     Customer_rating Cost_of_the_Product
                  0                   0                   0                   0                   0
    Prior_purchases  Product_importance              Gender    Discount_offered       Weight_in_gms
                  0                   0                   0                   0                   0
Reached.on.Time_Y.N
                  0
> BlankValue = function (x) {sum(x=="") }
> apply(dat, 2, BlankValue)
    Warehouse_block    Mode_of_Shipment Customer_care_calls     Customer_rating Cost_of_the_Product
                  0                   0                   0                   0                   0
    Prior_purchases  Product_importance              Gender    Discount_offered       Weight_in_gms
                  0                   0                   0                   0                   0
Reached.on.Time_Y.N
                  0
```

Для перегляду кількості унікальних значень та дублікатів скористуємось функціями *UniqueValue* та *DuplicateValue*.

```
> UniqueValue = function (x) {length(unique(x)) }
> apply(dat, 2, UniqueValue)
    Warehouse_block    Mode_of_Shipment Customer_care_calls     Customer_rating Cost_of_the_Product
                  5                   3                   6                   5                 215
    Prior_purchases  Product_importance              Gender    Discount_offered       Weight_in_gms
                  8                   3                   2                  65                4034
Reached.on.Time_Y.N
                  2
> DuplicateValue= function(x) {sum(duplicated(x)) }
> apply(dat, 2, DuplicateValue)
    Warehouse_block    Mode_of_Shipment Customer_care_calls     Customer_rating Cost_of_the_Product
              10994               10996               10993               10994               10784
    Prior_purchases  Product_importance              Gender    Discount_offered       Weight_in_gms
              10991               10996               10997               10934                6965
Reached.on.Time_Y.N
              10997
```

Для перевірки набору даних на аномальні значення побудуємо гістограми значень атрибутів.

Рисунок 3.7 – Гістограми змінних *Customer_rating* та *Prior_purchases*

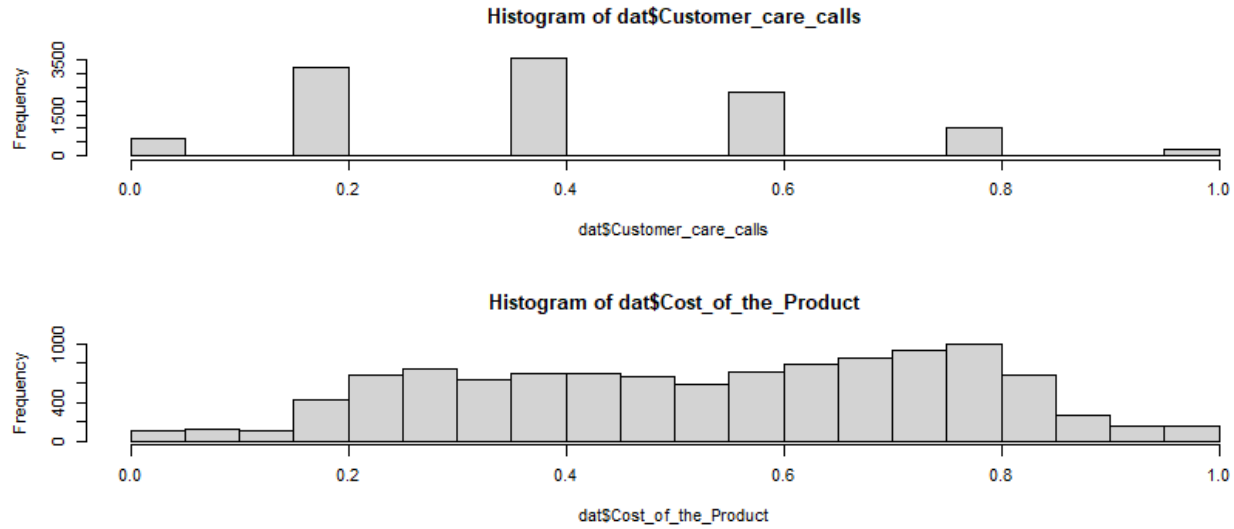


Рисунок 3.8 – Гістограми змінних *Cost_of_the_Product* та *Customer_care_calls*

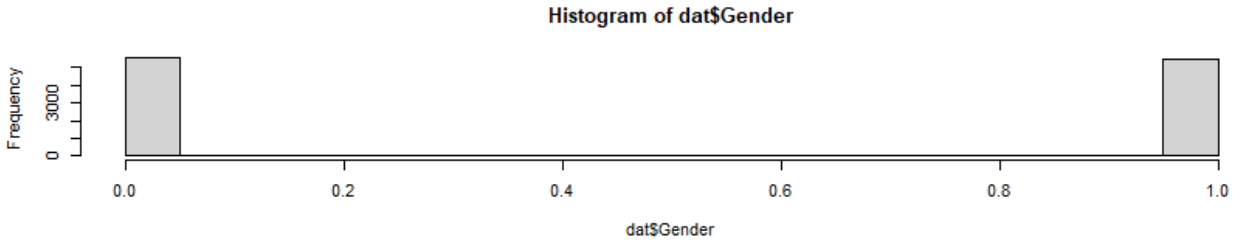


Рисунок 3.9 – Гістограма змінної *Gender*

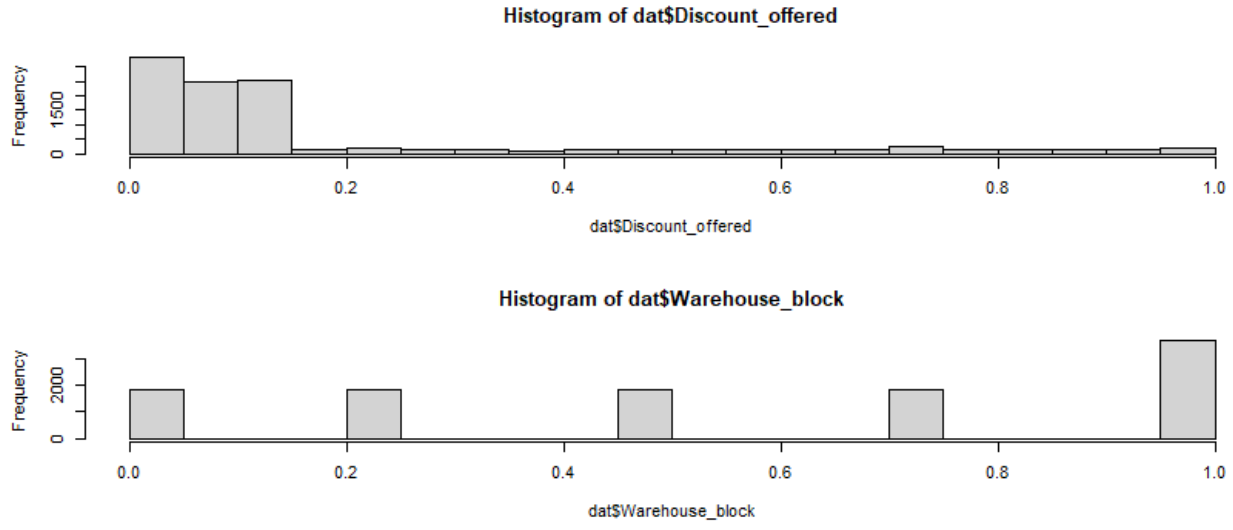


Рисунок 3.10 – Гістограми змінних *Discount_offered* та *Warehouse_block*

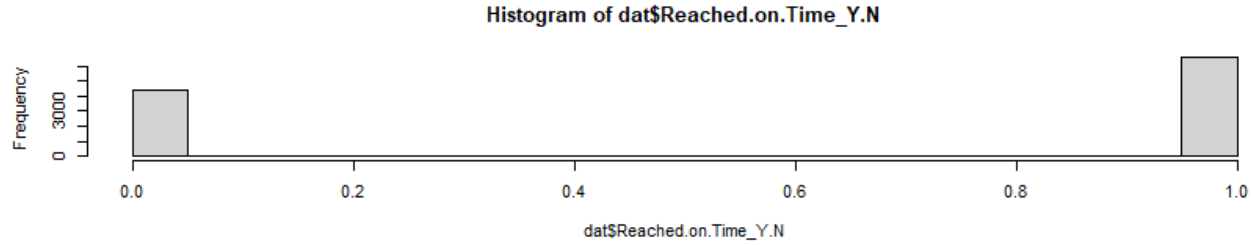


Рисунок 3.11 – Гістограми змінної *Reached.on.Time_Y.N*

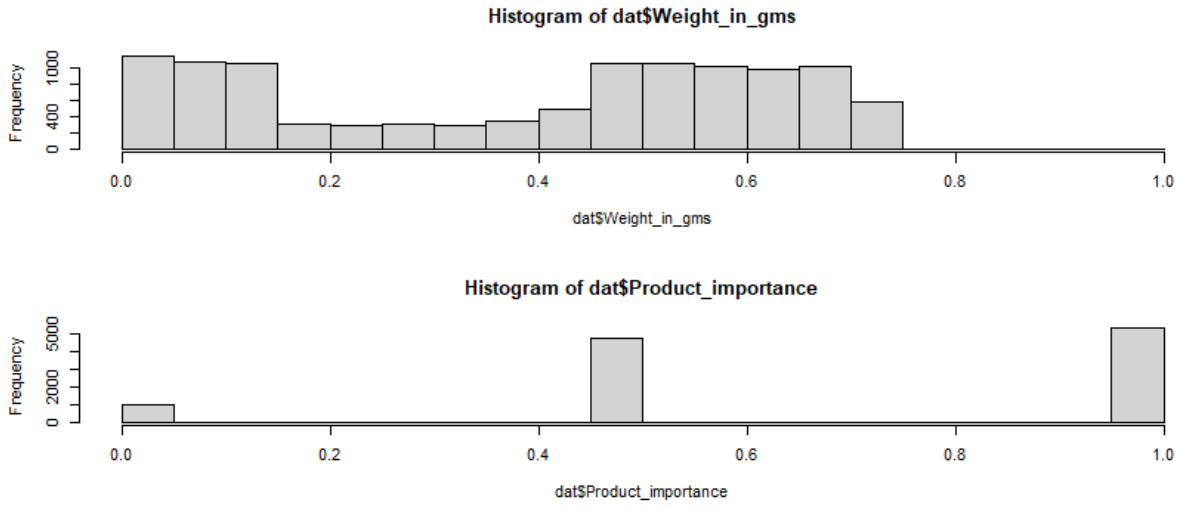


Рисунок 3.12 – Гістограми змінних *Weight_in_gms* та *Product_Importance*

З наведених гістограм видно, що аномальні значення у наборі даних відсутні.

Перш ніж створювати модель за даними, необхідно визначити, яким чином незалежні змінні пов'язані із залежною змінною та один з одним. Швидку відповідь це питання дає матриця кореляції. Вона показує кореляцію кожної пари змінних із заданого набору.

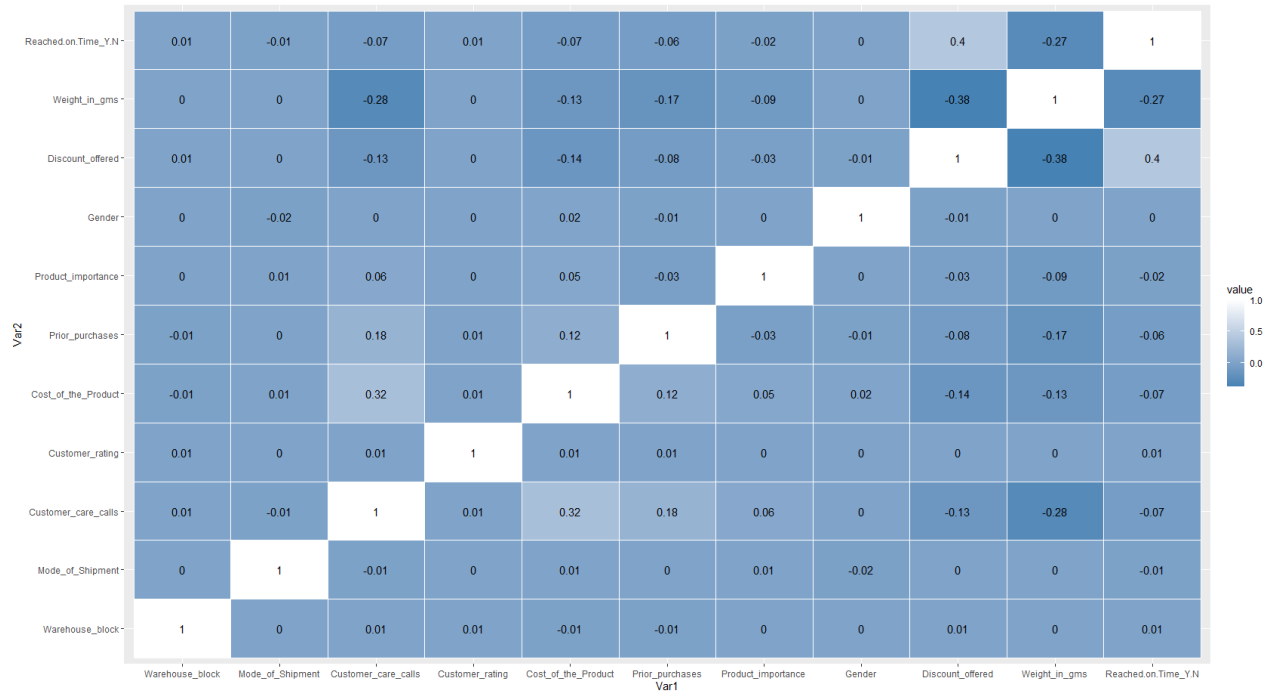


Рисунок 3.13 – Матриця кореляції

Кореляція часто інтерпретується залежно від контексту. Однак існує низка емпіричних правил, які використовуються для інтерпретації інтенсивності кореляції. Найбільш поширений практично метод надає статус «слабкий» значенням від 0,1 до 0,3, «помірний» – значенням у діапазоні від 0,3 до 0,5, і «сильний» для значень вище 0,5 (це також стосується до аналогічних діапазонів негативних кореляцій). З наведеної вище матриці видно, що *Customer_care_calls* має сильний зв’язок із змінною *Cost_of_the_Product*, а *Discount_offered* має сильний зв’язок із змінною *Weight_in_gms*.

```
> prop.table(table(dat$Reached.on.Time_Y.N))

       no       yes
0.5966906 0.4033094
```

Прорахуємо аналіз ризиків. Вектор *Reached.on.Time_Y.N* вказує на те, чи змогла компанія вчасно чи з затримкою доставити певний товар покупцю. Загалом приблизно 60 % товару із цього набору даних було доставлено невчасно. Таким

чином, спостерігається дисбаланс класів, що виникає, коли переважна більшість записів набору даних відносяться до одного класу.

Отже, з'ясовано, що в обраному для досліджень наборі даних немає нечислових, відсутніх чи аномальних значень в жодному атрибуті. Проведено заміну та нормалізацію даних. Таким чином набор готовий до етапу моделювання.

Підготовлений для моделювання набір даних поділяється на дві частини: *тренувальний набір* для побудови базової моделі класифікатора та *тестовий набір* для оцінки ефективності моделі нових даних. Використовується 90% даних для навчання та 10% – для тестування, що дасть 1999 записів для моделювання нових претендентів. Процедура поділу даних на тренувальний та тестовий набори називається методом відкладених даних.

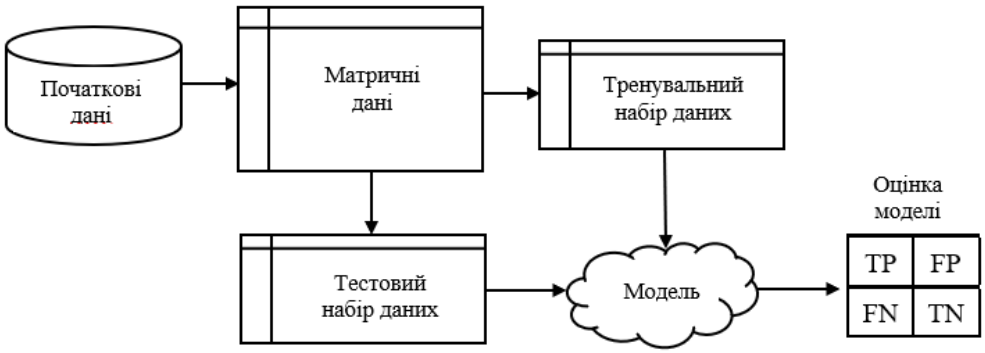


Рисунок 3.14 – Схема метода відкладених даних

Вихідний (початковий) набір даних сервісів доставки не впорядкований випадково, тому відібрати перші по черзі 90% записів було б неправильно. Тому модель навчатиметься на випадковій вибірці логістичних даних. Випадкова вибірка – це процес, який випадково вибирає підмножину записів. У R для виконання випадкової вибірки використовується функція *sample()*. Установка початкового значення з допомогою функції *set.seed()* гарантує, що й потім аналіз повторюватися, буде отримано той самий результат.

Отриманий в результаті об'єкт train_sample являє собою вектор із 900 випадкових цілих чисел:

```
> set.seed(1234)
> train_sample<-sample(10999,9000)
> str(train_sample)
 int [1:9000] 7452 8016 7162 8086 7269 9196 623 10885 934 2948
```

Для кращої інтерпретації вихідних результатів результуючої змінної, виконано заміну значень *Reached.on.Time_Y.N*, де 0 → yes (доставлено вчасно), 1 → no (доставлено з затримкою).

```
> dat$Reached.on.Time_Y.N = ifelse(dat$Reached.on.Time_Y.N == 0, 'yes', 'no')
> dat$Reached.on.Time_Y.N <- as.factor(dat$Reached.on.Time_Y.N)
```

Використовуючи цей вектор для вибору рядків із кредитних даних, підготовлений набір даних поділяється (9:1) на 90% тренувальних та 10% тестових даних. Якщо рандомізація виконана правильно, то в кожному наборі даних (тренувальному та тестовому) має бути приблизно 60% доставлених с затримкою товарів. Це відповідає співвідношенню класів у вихідному наборі даних.

Аналіз співвідношення класів «похибка» пов'язана з вихідним дисбалансом між класами *"no"* та *"yes"*.

```
> prop.table(table(delivery_train$Reached.on.Time_Y.N))

    no    yes
0.5985 0.4015
> prop.table(table(delivery_test$Reached.on.Time_Y.N))

      no      yes
0.591864 0.408136
```

Оскільки в тренувальному та тестовому наборах даних кількість вчасно не доставлених товарів виявилася приблизно однаковою, можна приступити до побудови моделі.

Висновки до розділу 3

1. Мета моделі – виявити чинники, пов'язані з ризиком невчасної доставки товару клієнту.

2. Набір даних, що наданий міжнародною компанією, яка хоче отримати дані сервісу доставки зі своєї бази даних клієнтів та використаний для побудови моделі, містить 10999 спостережень та 12 змінних.

3. До нечислових або факторних змінних відносяться атрибути: Warehouse_block, Mode_of_Shipment, Product_Importance та Gender. Змінні ID, Customer_care_calls, Customer_rating, Cost_of_the_Product, Prior_purchases,

Discount_offered та Weight_in_gms, а також результат класифікації атрибут Reached.on.Time_Y.N.

4. Перетворено факторних змінні Warehouse_block, Mode_of_Shipment, Product_Importance та Gender на числові, оскільки для більшості моделей машинного навчання на вхід подаються числові значення.

5. Виконано нормалізацію даних за допомогою мінімаксного методу.

6. При перевірці з'ясовано, що в обраному для досліджень наборі даних немає нечислових чи відсутніх значень в жодному атрибуті.

7. Аномальні значення у наборі даних відсутні.

8. З матриці кореляції видно, що Customer_care_calls має сильний зв'язок із змінною Cost_of_the_Product, а Discount_offered має сильний зв'язок із змінною Weight_in_gms.

9. Вектор Reached.on.Time_Y.N вказує на те, чи змогла компанія вчасно чи з затримкою доставити певний товар покупцю. Загалом приблизно 60 % товару із цього набору даних було доставлено невчасно.

10. Підготовлений для моделювання набір даних поділено на дві частини: тренувальний набір для побудови базової моделі класифікатора та тестовий набір для оцінки ефективності моделі нових даних. Використано 90% даних для навчання та 10% – для тестування, що дало 1999 записів для моделювання нових претендентів.

4 ОЦІНЮВАННЯ ЯКОСТІ РОБОТИ БАЗОВИХ КЛАСИФІКАТОРІВ ТА ДВОРІВНЕНОГО АНСАМБЛЮ МОДЕЛІ

4.1 Результати застосування простих класифікаторів для вирішення поставленої задачі

У якості базових моделей обрано моделі: дерева рішень (Decision Tree), Наївний Баєсів класифікатор (NB), лінійний дискримінантний аналіз (LDA), квадратичний дискримінантний аналіз (QDA), логістична регресія (LR), метод опорних векторів (SVM), метод найближчого сусіда (KNN), штучні нейронні мережі (ANN) та модель випадкового лісу (RF).

Тренувальний набір *delivery_train* містить 9000 спостережень, а також відгуки (*Reached.on.Time_Y.N*) кожного спостереження. Створюючи модель, повідомляємо алгоритм, яка змінна є відгуком. Алгоритм класифікації використовує відгук для навчання моделі, досліджуючи взаємозв'язки між змінними предиктора (будь-яким з 10 атрибутів) та змінною відгуком (*Reached.on.Time_Y.N*).

Набір тестів *delivery_test* містить інші дані (тобто частину, не включену до тренувального набору, 1999 спостережень).

Дерева рішень. Для створення базової моделі дерева рішень зі змінною *Reached.on.Time_Y.N* як відгук та всіх інших змінних як предикторів використовують пакет *party*:

```
> model_tr_1<-ctree(Reached.on.Time_Y.N~., delivery_train)
```

Інформацію про створену модель отримаємо командою summary:

```
> summary(model_tr_1)
     Length      Class       Mode
          1 BinaryTree         S4
> model_tr_1

        Conditional inference tree with 9 terminal nodes

Response:  Reached.on.Time_Y.N
Inputs:  Warehouse_block, Mode_of_Shipment, Customer_care_calls, Customer_rating, Cost_of_the_Product,
Number of observations:  9000

1) Discount_offered <= 0.140625; criterion = 1, statistic = 1412.832
  2) Weight_in_gms <= 0.4558072; criterion = 1, statistic = 49.072
    3) Cost_of_the_Product <= 0.5; criterion = 1, statistic = 72.857
      4) Weight_in_gms <= 0.4384222; criterion = 1, statistic = 23.363
```

Стовпець Class повідомляє, що створено дерево рішень. Об'єкт model_tr_1 містить дерево рішень C5.0. Основні відомості про це дерево:

Візуалізація моделі виконується за допомогою коду:

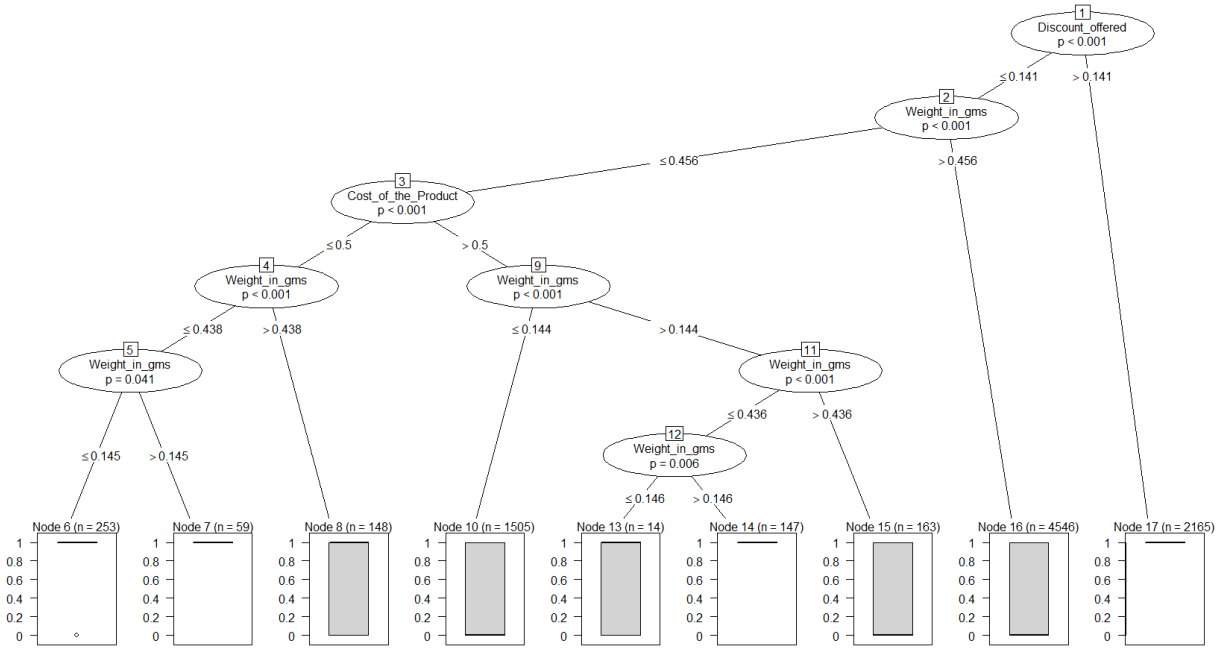


Рисунок 4.1 – Візуалізація дерева рішень для класифікації

Щоб з'ясувати, чи добре навчена модель, необхідно звірити її з даними навчання. Результат отримуємо як таблиці.

```
> table(predict(model_tr_1), delivery_train $Reached.on.Time_Y.N)

       no  yes
  no  2666  120
  yes 2715 3499
> (2715+120)/9000
[1] 0.315
```

Результат показує, що помилка (або рівень помилкової класифікації) становить 2835 із 9000, або 31,5 %. Таким чином, точність класифікації на тренувальних даних – 68,5 %.

Наступний етап – читання таблиці. Правильні прогнози – це прогнози, номери стовпців та рядків яких однакові. Ці результати відображаються у вигляді діагональної лінії від верхнього лівого до нижнього правого; наприклад, [1,1], [2,2] - правильні передбачення. Таким чином, невчасно доставлені товари правильно передбачені 2666 рази, у той же час неправильно класифіковані 120 разів. Товари, які були доставлені вчасно правильно передбачені 3499 разів, неправильно – 2715 рази. У результаті 120 реальних значень “*no*” були помилково класифіковані як

“*yes”* (хибно-позитивні спрацьовування), а 2715 значення “*yes”* були помилково класифіковані як “*no”* (хибно-негативні спрацьовування).

Враховуючи тенденцію дерев рішень до перенавчання на тренувальних даних, наведена частота помилок, отримана на основі ефективності тренувальних даних, може бути надмірно оптимістичною. Тому особливо важливо продовжити оцінювати алгоритм, застосувавши дерево рішень до тестового набору даних.

Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_tr<-predict(model_tr_1,newdata= delivery_test)
> delivery_test$testPrediction_tr <- testPrediction_tr
> table(testPrediction_tr, delivery_test $Reached.on.Time_Y.N)

testPrediction_tr  no yes
               no  598  13
               yes 584 804
> (584+13)/1999
[1] 0.2986493
```

Результат показує, що помилка становить 597 зі 1999, або 29,8 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 70,2 %.

Вектор *testPrediction_tr* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()* з пакета *gmodels*. Якщо надати параметрам *prop.c* і *prop.r* значення FALSE, то відсотковий вміст стовпців і рядків буде видалено з таблиці. Відсоток (prop.t), що залишився, вказує частку записів в комірках від загальної кількості записів. Наведемо розширену матрицю невідповідностей:

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_tr,prop.chisq =
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))


   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       598 |       584 |      1182 |
             |     0.299 |     0.292 |           |
-------------|-----------|-----------|-----------|
         yes |        13 |       804 |       817 |
             |     0.007 |     0.402 |           |
-------------|-----------|-----------|-----------|
Column Total |       611 |      1388 |      1999 |
-------------|-----------|-----------|-----------|
```

Наведемо матрицю невідповідностей, крос-таблицю для дерева рішень.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_tr,dnn=c('actua


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       598 |       584 |      1182 |
             |   155.102 |    68.276 |           |
             |     0.506 |     0.494 |     0.591 |
             |     0.979 |     0.421 |           |
             |     0.299 |     0.292 |           |
-------------|-----------|-----------|-----------|
         yes |        13 |       804 |       817 |
             |   224.395 |    98.779 |           |
             |     0.016 |     0.984 |     0.409 |
             |     0.021 |     0.579 |           |
             |     0.007 |     0.402 |           |
-------------|-----------|-----------|-----------|
Column Total |       611 |      1388 |      1999 |
             |     0.306 |     0.694 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 598 вчасно було доставлено, а в 804 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 598 з 1182 реальних невчасних доставок, тобто 50,6%. На жаль, цей тип помилок є потенційно дуже дорогим, оскільки при кожній затримці доставки компанія втрачає клієнтів.

Наївний Баєсів класифікатор (NB). Для створення базової моделі NB зі змінною *Reached.on.Time_Y.N* як відгук та всіх інших змінних як предикторів використовують пакет *naivebayes*:

```
> model_tr_NB <- naive_bayes(Reached.on.Time_Y.N ~ ., data = delivery_train, useker$
```

Інформацію про створену модель отримаємо командою summary:

```
> summary(model_tr_NB)

================================== Naive Bayes =============

- Call: naive_bayes.formula(formula = Reached.on.Time_Y.N ~
- Laplace: 0
- Classes: 2
- Samples: 9000
- Features: 10
- Conditional distributions:
    - KDE: 10
- Prior probabilities:
    - no: 0.5979
    - yes: 0.4021
```

Тренувальна модель містить 9000 спостережень, 2 класи розподілу: вчасно та невчасно доставлений товар, оцінка щільності ядра (KDE): 10, попередні ймовірності no – 59 %, yes – 41 %. Основні відомості про цю модель:

```
> model_tr_NB
 A priori probabilities:

       no       yes
0.5978889 0.4021111

-------------------------------------------------

 Tables:

-------------------------------------------------
 ::: Warehouse_block::no (KDE)
-------------------------------------------------

Call:
        density.default(x = x, na.rm = TRUE)

Data: x (5381 obs.);    Bandwidth 'bw' = 0.06031

       x                 y
 Min.   :-0.1809   Min.   :0.01225
 1st Qu.: 0.1595   1st Qu.:0.35688
 Median : 0.5000   Median :0.66242
 Mean   : 0.5000   Mean   :0.73312
 3rd Qu.: 0.8405   3rd Qu.:1.01519
 Max.   : 1.1809   Max.   :2.22502
```

Візуалізація моделі NB виконується за допомогою команди *plot*:

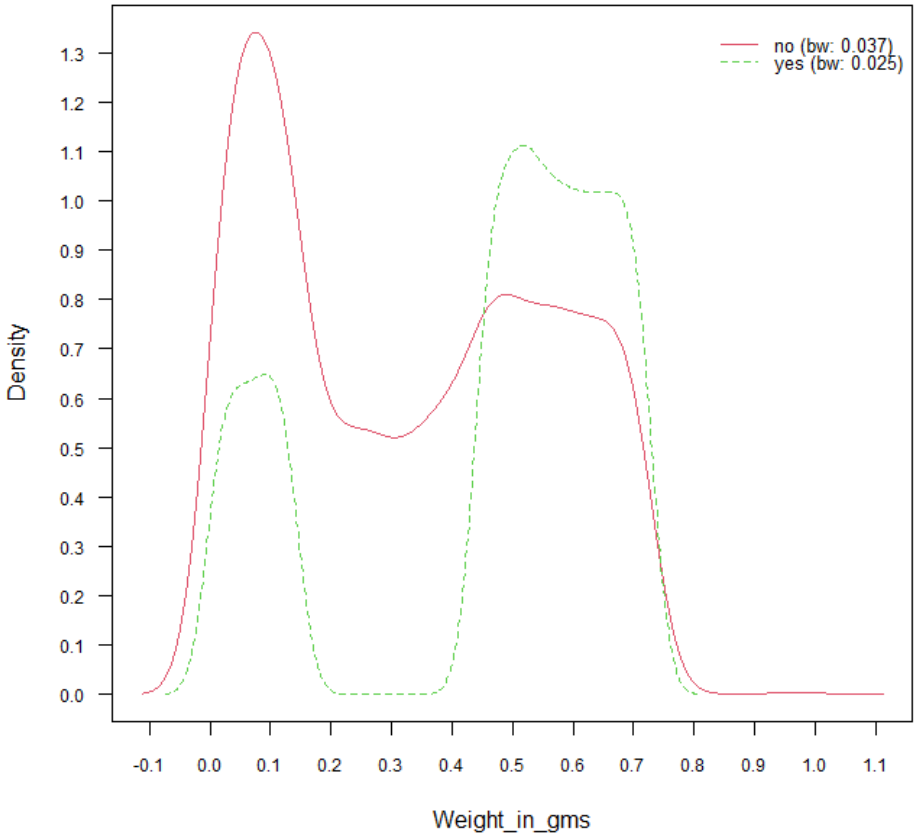


Рисунок 4.2 – Візуалізація моделі NB для класифікації

Щоб з'ясувати, чи добре навчена модель, необхідно звірити її з даними навчання. Результат отримуємо як таблиці.

```
> table(predict(model_tr_NB), delivery_train $Reached.on.Time_Y.N)

        no  yes
  no  3098  653
  yes 2283 2966
> (2283+653)/9000
[1] 0.3262222
```

Результат показує, що помилка (або рівень помилкової класифікації) становить 2936 із 9000, або 32,6%. Таким чином, точність класифікації на тренувальних даних – 67,4%.

З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 3098 рази, у той же час неправильно класифіковані 653 разів. Товари, які були доставлені вчасно правильно передбачені 2966 разів, неправильно – 2283 рази. У результаті 653 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 2283 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_NB<-predict(model_tr_NB,newdata= delivery_test)
> delivery_test$testPrediction_NB <- testPrediction_NB
> table(testPrediction_NB, delivery_test $Reached.on.Time_Y.N)

testPrediction_NB  no yes
               no  693 163
               yes 489 654
> (489+163)/1999
[1] 0.3261631
```

Результат показує, що помилка становить 652 зі 1999, або 32,6 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 67,4 %.

Вектор *testPrediction_NB* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_NB,
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted')


   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       693 |       489 |      1182 |
             |     0.347 |     0.245 |           |
-------------|-----------|-----------|-----------|
         yes |       163 |       654 |       817 |
             |     0.082 |     0.327 |           |
-------------|-----------|-----------|-----------|
Column Total |       856 |      1143 |      1999 |
-------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для наївного Баєсівського класифікатору.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_NB,


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       693 |       489 |      1182 |
             |    68.978 |    51.658 |           |
             |     0.586 |     0.414 |     0.591 |
             |     0.810 |     0.428 |           |
             |     0.347 |     0.245 |           |
-------------|-----------|-----------|-----------|
         yes |       163 |       654 |       817 |
             |    99.795 |    74.737 |           |
             |     0.200 |     0.800 |     0.409 |
             |     0.190 |     0.572 |           |
             |     0.082 |     0.327 |           |
-------------|-----------|-----------|-----------|
Column Total |       856 |      1143 |      1999 |
             |     0.428 |     0.572 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 654 вчасно було доставлено, а в 693 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 693 з 1182 реальних невчасних доставок, тобто 58,6%. На жаль, цей тип помилок є потенційно дуже дорогим, оскільки при кожній затримці доставки компанія втрачає клієнтів.

Квадратичний дискримінантний аналіз (QDA). Для створення базової моделі QDA зі змінною *Reached.on.Time_Y.N* як відгук та всіх інших змінних як предикторів використовують пакет *MASS*:

```
> model_tr_QDA = qda(Reached.on.Time_Y.N ~ ., data = delivery_train)
```

Щоб побачити інформацію про створену модель, скористуємось командою summary:

```
> summary(model_tr_QDA)
        Length Class  Mode
prior     2    -none- numeric
counts    2    -none- numeric
means    20    -none- numeric
scaling 200    -none- numeric
ldet      2    -none- numeric
lev       2    -none- characte
N         1    -none- numeric
call      3    -none- call
terms     3    terms  call
xlevels   0    -none- list
```

Модель QDA містить параметри *prior(num), counts(num), means(num), scalling (num), ldet(num), lev(char), N(num), call(call), terms(call) ma xlevels(list).*

Основні відомості про цю модель:

```
> model_tr_QDA
Call:
qda(Reached.on.Time_Y.N ~ ., data = delivery_train)

Prior probabilities of groups:
       no       yes
0.5978889 0.4021111

Group means:
    Warehouse_block Mode_of_Shipment Customer_care_calls Customer_rating
no        0.5828842        0.4973983           0.4001115       0.4986062
yes       0.5808925        0.5012434           0.4330478       0.4917104
    Cost_of_the_Product Prior_purchases Product_importance    Gender
no            0.5205647       0.1874884          0.6930868 0.5002788
yes           0.5558243       0.2103827          0.7054435 0.5009671
    Discount_offered Weight_in_gms
no        0.27681309     0.3306767
yes       0.07128609     0.4596332
```

Модель вказує на співвідношення між атрибутами набору та класами "no" та "yes" результуючої змінної Reached.on.Time_Y.N.

Щоб з'ясувати, чи добре навчена модель, необхідно звірити її з даними навчання. Результат отримуємо як таблиці.

```
> predmodel.train.qda = predict(model_tr_QDA, data=delivery_train)
> table(predmodel.train.qda$class, delivery_train $Reached.on.Time_Y.N)

       no  yes
  no  2411   85
  yes 2970 3534
> (2970+85)/9000
[1] 0.3394444
```

Результат показує, що помилка становить 3055 із 9000, або 34 %. Таким чином, точність класифікації на тренувальних даних – 63 %. З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 2411 рази, у той же час неправильно класифіковані 85 разів. Товари, які були доставлені вчасно правильно передбачені 3534 разів, неправильно – 2970 рази. У результаті 85

реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 2970 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_QDA<-predict(model_tr_QDA,delivery_test)
> delivery_test$testPrediction_QDA <- testPrediction_QDA$class
> table(testPrediction_QDA$class, delivery_test $Reached.on.Time_Y.N)

      no yes
  no  531  15
  yes 651 802
> (651+15)/1999
[1] 0.3331666
```

Результат показує, що помилка становить 666 зі 1999, або 33,3 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 66,7 %.

Вектор *testPrediction_QDA* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_QDA
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted')


   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999

              | predicted
       actual |        no |       yes | Row Total |
--------------|-----------|-----------|-----------|
           no |       531 |       651 |      1182 |
              |     0.266 |     0.326 |           |
--------------|-----------|-----------|-----------|
          yes |        15 |       802 |       817 |
              |     0.008 |     0.401 |           |
--------------|-----------|-----------|-----------|
Column Total |       546 |      1453 |      1999 |
--------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для квадратичного дискримінантного аналізу.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_QDA$

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       531 |       651 |      1182 |
             |   134.204 |    50.431 |           |
             |     0.449 |     0.551 |     0.591 |
             |     0.973 |     0.448 |           |
             |     0.266 |     0.326 |           |
-------------|-----------|-----------|-----------|
         yes |        15 |       802 |       817 |
             |   194.161 |    72.961 |           |
             |     0.018 |     0.982 |     0.409 |
             |     0.027 |     0.552 |           |
             |     0.008 |     0.401 |           |
-------------|-----------|-----------|-----------|
Column Total |       546 |      1453 |      1999 |
             |     0.273 |     0.727 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 802 вчасно було доставлено, а в 531 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 531 з 1182 реальних невчасних доставок, тобто 45%. На жаль, цей тип помилок є потенційно дуже дорогим, оскільки при кожній затримці доставки компанія втрачає клієнтів.

Лінійний дискримінантний аналіз (LDA). Для створення базової моделі LDA зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *MASS*:

```
> model_tr_LDA = lda(Reached.on.Time_Y.N ~ ., data = delivery_train)
```

Щоб побачити інформацію про створену модель, скористуємось командою summary:

Модель LDA містить параметри *prior(num), counts(num), means(num), scalling (num), ldet(num), lev(char), N(num), call(call), terms(call) ma xlevels(list).*

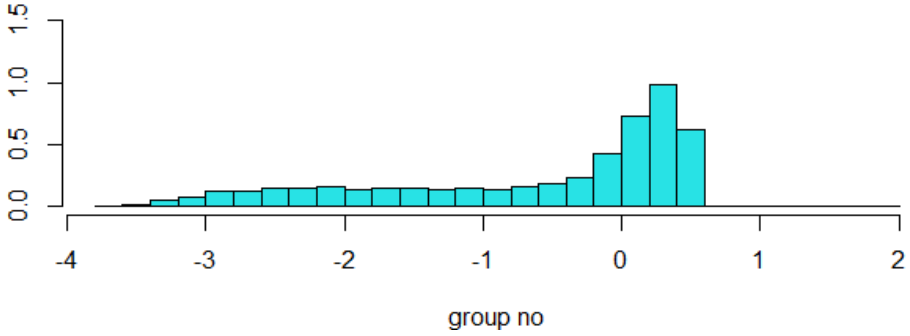
```
> summary(model_tr_LDA)
        Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means   20     -none- numeric
scaling 10     -none- numeric
lev      2     -none- character
svd      1     -none- numeric
N        1     -none- numeric
call     3     -none- call
terms    3     terms  call
xlevels  0     -none- list
```

Основні відомості про цю модель:

```
> model_tr_LDA
Call:
lda(Reached.on.Time_Y.N ~ ., data = delivery_train)

Prior probabilities of groups:
       no       yes
0.5978889 0.4021111

Group means:
    Warehouse_block Mode_of_Shipment Customer_care_calls Customer_rating
no        0.5828842        0.4973983           0.4001115       0.4986062
yes       0.5808925        0.5012434           0.4330478       0.4917104
    Cost_of_the_Product Prior_purchases Product_importance    Gender
no            0.5205647       0.1874884          0.6930868 0.5002788
yes           0.5558243       0.2103827          0.7054435 0.5009671
    Discount_offered Weight_in_gms
no        0.27681309     0.3306767
yes       0.07128609     0.4596332

Coefficients of linear discriminants:
                            LD1
Warehouse_block     -0.008891578
Mode_of_Shipment     0.060953199
Customer_care_calls  0.660665966
Customer_rating     -0.074268487
Cost_of_the_Product  0.320965428
Prior_purchases      0.658774711
Product_importance   0.201330341
Gender              -0.032910266
Discount_offered    -3.201794971
Weight_in_gms        1.919534884
```

Модель вказує на співвідношення між атрибутами набору та класами "no" та "yes" результуючої змінної Reached.on.Time_Y.N, а також на коефіцієнт лінійної дискримінації для кожного атрибуту набору.

Візуалізація моделі LDA виконується за допомогою команди *ldahist.*



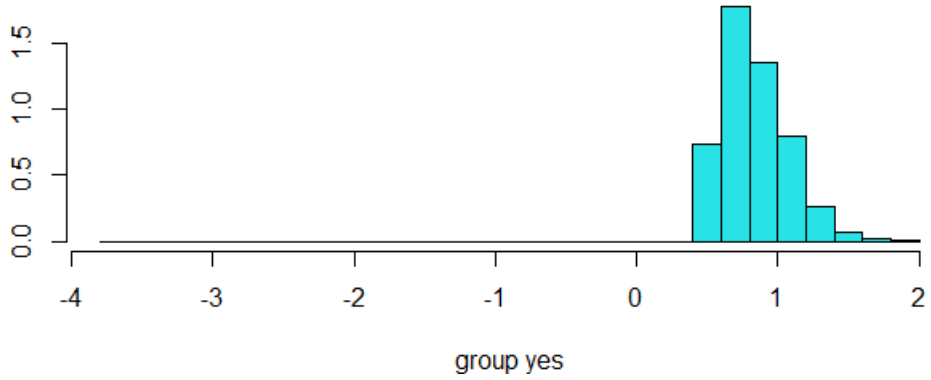


Рисунок 4.3 – Візуалізація результатів моделі LDA для класифікації

Щоб з'ясувати, чи добре навчена модель, необхідно звірити її з даними навчання. Результат отримуємо як таблиці.

```
> predmodel.train.lda = predict(model_tr_LDA, data=delivery_train)
> table(predmodel.train.lda$class,delivery_train $Reached.on.Time_Y.N)

        no  yes
  no  3762 1508
  yes 1619 2111
> (1619+1508)/9000
[1] 0.3474444
```

Результат показує, що помилка становить 3127 із 9000, або 34,7 %. Таким чином, точність класифікації на тренувальних даних – 65,3 %.

З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 3762 рази, у той же час неправильно класифіковані 1508 разів. Товари, які були доставлені вчасно правильно передбачені 2111 разів, неправильно – 1619 рази. У результаті 1508 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 1619 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_LDA<-predict(model_tr_LDA,delivery_test)
> delivery_test$testPrediction_LDA <- testPrediction_LDA$class
> table(testPrediction_LDA$class,delivery_test $Reached.on.Time_Y.N)

       no yes
  no  806 337
  yes 376 480
> (376+337)/1999
[1] 0.3566783
```

Результат показує, що помилка становить 713 зі 1999, або 35,7 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 64,3 %.

Вектор *testPrediction_LDA* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_LDA
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted')

   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  1999

             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       806 |       376 |      1182 |
             |     0.403 |     0.188 |           |
-------------|-----------|-----------|-----------|
         yes |       337 |       480 |       817 |
             |     0.169 |     0.240 |           |
-------------|-----------|-----------|-----------|
Column Total |      1143 |       856 |      1999 |
-------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для лінійного дискримінантного аналізу.

```
             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       806 |       376 |      1182 |
             |    25.063 |    33.466 |           |
             |     0.682 |     0.318 |     0.591 |
             |     0.705 |     0.439 |           |
             |     0.403 |     0.188 |           |
-------------|-----------|-----------|-----------|
         yes |       337 |       480 |       817 |
             |    36.260 |    48.417 |           |
             |     0.412 |     0.588 |     0.409 |
             |     0.295 |     0.561 |           |
             |     0.169 |     0.240 |           |
-------------|-----------|-----------|-----------|
Column Total |      1143 |       856 |      1999 |
             |     0.572 |     0.428 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 480 вчасно було доставлено, а в 806 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 806 з 1182 реальних невчасних доставок,

тобто 68,1%. Цей тип помилок є потенційно дуже дорогим, оскільки при кожній затримці доставки компанія втрачає клієнтів.

Логістична регресія (LR). Для створення базової моделі LR зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *stats*:

```
> model_tr_LR = glm(Reached.on.Time_Y.N ~ ., data = delivery_train, family = binomial())
```

Щоб побачити інформацію про створену модель, скористуємось командою summary:

```
> summary(model_tr_LR)

Call:
glm(formula = Reached.on.Time_Y.N ~ ., family = binomial(), data = delivery_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8938  -1.0873  -0.1336   1.0822   1.7104

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -0.8471005  0.1506517  -5.623 1.88e-08 ***
Warehouse_block     -0.0002918  0.0644254  -0.005 0.996386
Mode_of_Shipment     0.0326739  0.0846878   0.386 0.699633
Customer_care_calls  0.5563872  0.1189883   4.676 2.93e-06 ***
Customer_rating     -0.0612608  0.0681363  -0.899 0.368604
Cost_of_the_Product  0.4062080  0.1173542   3.461 0.000537 ***
Prior_purchases      0.6230671  0.1335686   4.665 3.09e-06 ***
Product_importance   0.1938015  0.0769228   2.519 0.011754 *
Gender              -0.0397852  0.0481911  -0.826 0.409047
Discount_offered    -7.0873926  0.3122025 -22.701  < 2e-16 ***
Weight_in_gms        1.5924022  0.1211853  13.140  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 12129.5  on 8999  degrees of freedom
Residual deviance:  9845.9  on 8989  degrees of freedom
AIC: 9867.9

Number of Fisher Scoring iterations: 6
```

Модель LR містить параметри оцінки, середнє квадратичне відхилення середнього арифметичного, стандартизована z оцінка та p-значення імовірності кожного атрибута моделі, а також загальне нульове та остаточне відхилення. Серед основних відомостей: ступінь вільності, нульове та остаточне відхилення.

Основні відомості про цю модель:

```
> model_tr_LR

Call:  glm(formula = Reached.on.Time_Y.N ~ ., family = binomial(), data = delivery_train)

Coefficients:
        (Intercept)      Warehouse_block     Mode_of_Shipment  Customer_care_calls
          -0.8471005           -0.0002918            0.0326739            0.5563872
    Customer_rating  Cost_of_the_Product      Prior_purchases   Product_importance
          -0.0612608            0.4062080            0.6230671            0.1938015
              Gender     Discount_offered       Weight_in_gms
          -0.0397852           -7.0873926            1.5924022

Degrees of Freedom: 8999 Total (i.e. Null);  8989 Residual
Null Deviance:      12130
Residual Deviance: 9846         AIC: 9868
```

Візуалізація моделі LR виконується за допомогою команди *plot*.

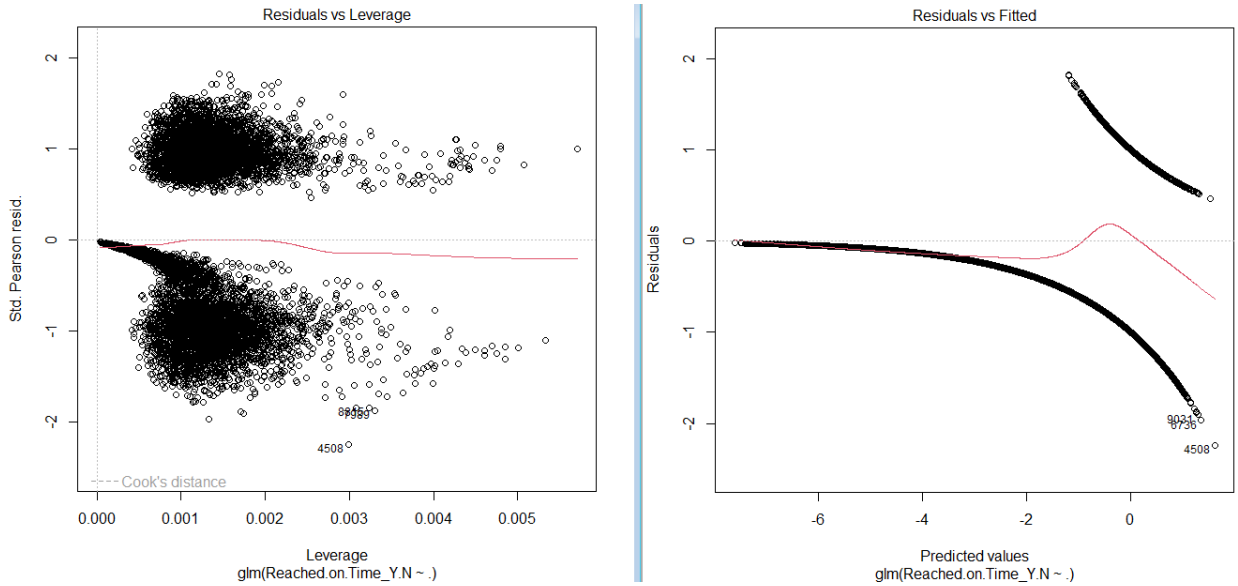


Рисунок 4.4 – Візуалізація результатів моделі LR для класифікації

Щоб з'ясувати, чи добре навчена модель, необхідно звірити її з даними навчання. Результат отримуємо як таблиці.

```
> pr_tr_LR = predict(model_tr_LR, type="response")
> pr_tr_LR = ifelse(pr_tr_LR > 0.5, 1, 0)
> table(pr_tr_LR,delivery_train $Reached.on.Time_Y.N)

pr_tr_LR   no  yes
       0 3672 1550
       1 1709 2069
> (1709+1550)/9000
[1] 0.3621111
```

Результат показує, що помилка становить 3259 із 9000, або 36,2 %. Таким чином, точність класифікації на тренувальних даних – 63,8 %.

З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 3672 рази, у той же час неправильно класифіковані 1550 разів. Товари, які були доставлені вчасно правильно передбачені 2069 разів, неправильно – 1709

рази. У результаті 1550 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 1709 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_LR <- predict(model_tr_LR, newdata= delivery_test,typ
> delivery_test$testPrediction_LR <- testPrediction_LR
> testPrediction_LR = ifelse(testPrediction_LR > 0.5, 1, 0)
> test_LR<-table(testPrediction_LR,delivery_test $Reached.on.Time_Y.N,
> test_LR
      Predicted
Actual  no yes
     0 817 344
     1 365 473
> (365+344)/1999
[1] 0.3546773
```

Результат показує, що помилка становить 709 зі 1999, або 35,5 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 64,5 %.

Вектор *testPrediction_LR* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_LR,prop.chisq =
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  1999

              | predicted
       actual |         0 |         1 | Row Total |
--------------|-----------|-----------|-----------|
           no |       817 |       365 |      1182 |
              |     0.409 |     0.183 |           |
--------------|-----------|-----------|-----------|
          yes |       344 |       473 |       817 |
              |     0.172 |     0.237 |           |
--------------|-----------|-----------|-----------|
Column Total |      1161 |       838 |      1999 |
--------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для логістичної регресії.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_LR,

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


              | predicted
       actual |         0 |         1 | Row Total |
--------------|-----------|-----------|-----------|
           no |       817 |       365 |      1182 |
              |    24.810 |    34.372 |           |
              |     0.691 |     0.309 |     0.591 |
              |     0.704 |     0.436 |           |
              |     0.409 |     0.183 |           |
--------------|-----------|-----------|-----------|
          yes |       344 |       473 |       817 |
              |    35.894 |    49.729 |           |
              |     0.421 |     0.579 |     0.409 |
              |     0.296 |     0.564 |           |
              |     0.172 |     0.237 |           |
--------------|-----------|-----------|-----------|
 Column Total |      1161 |       838 |      1999 |
              |     0.581 |     0.419 |           |
--------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 473 вчасно було доставлено, а в 817 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 817 з 1182 реальних невчасних доставок, тобто 69,1%.

Метод опорних векторів (SVM). Для створення базової моделі SVM зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *e1071*:

```
> model_tr_SVM = svm(Reached.on.Time_Y.N ~ ., data = delivery_train,  cross = 10, kernel = "linear")
```

Параметр *cross* – це кількість k-кратної перехресної перевірки даних навчання, *kernel* – ядро для навчання та прогнозування.

Щоб побачити інформацію про створену модель, скористуємось командою summary:

```
> summary(model_tr_SVM)
Call:
svm(formula = Reached.on.Time_Y.N ~ ., data = delivery_train, cross = 10, kernel = "linear")
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
Number of Support Vectors:  6466

 ( 3234 3232 )

Number of Classes:  2

Levels:
 no yes

10-fold cross-validation on training data:

Total Accuracy: 66.12222
Single Accuracies:
 68.11111 66.44444 68.33333 65.44444 62.44444 66 64.66667 67.33333 64.33333 68.11111
```

Модель SVM має тип C-класифікації лінійного типу, кількість підтримуваних опорних векторів: 6466, кількість класів: 2, загальну точність на рівні 66,12% та одиночну точність по кожному атрибуту набору.

Візуалізація моделі SVM виконується за допомогою команди *plot*.

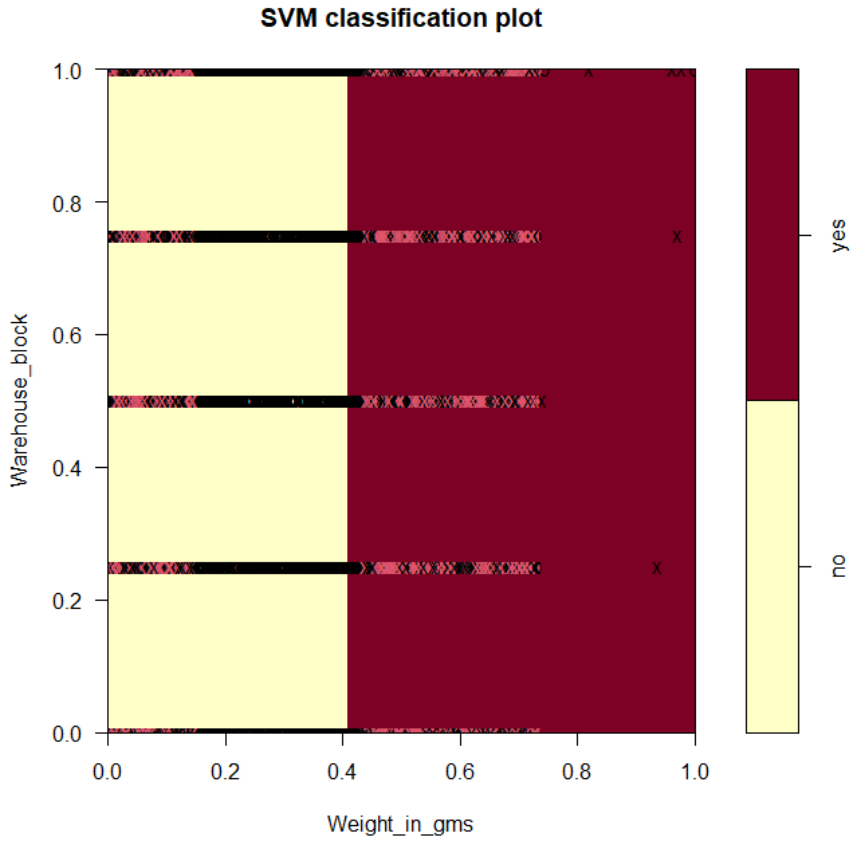


Рисунок 4.5 – Візуалізація результатів моделі SVM для класифікації

Щоб з'ясувати, чи добре навчена модель, необхідно звірити її з даними навчання. Результат отримуємо як таблиці.

```
> table(predict(model_tr_SVM),delivery_train $Reached.on.Time_Y.N)

        no  yes
  no  3273  941
  yes 2108 2678
> (2108+941)/9000
[1] 0.3387778
```

Результат показує, що помилка становить 3049 із 9000, або 33,8 %. Таким чином, точність класифікації на тренувальних даних – 66,2 %.

З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 3273 рази, у той же час неправильно класифіковані 941 разів. Товари, які були доставлені вчасно правильно передбачені 2678 разів, неправильно – 2108 рази. У результаті 941 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 2108 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_svm<-predict(model_tr_SVM,newdata= delivery_test)
> delivery_test$testPrediction_svm <- testPrediction_svm
> table(testPrediction_svm, delivery_test $Reached.on.Time_Y.N)

testPrediction_svm  no yes
                no  707 189
                yes 475 628
> (475+189)/1999
[1] 0.3321661
```

Результат показує, що помилка становить 664 зі 1999, або 33,2 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 66,8 %.

Вектор *testPrediction_SVM* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_svm,
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))


   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       707 |       475 |      1182 |
             |     0.354 |     0.238 |           |
-------------|-----------|-----------|-----------|
         yes |       189 |       628 |       817 |
             |     0.095 |     0.314 |           |
-------------|-----------|-----------|-----------|
Column Total |       896 |      1103 |      1999 |
-------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для методу опорних векторів.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_svm


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       707 |       475 |      1182 |
             |    59.267 |    48.144 |           |
             |     0.598 |     0.402 |     0.591 |
             |     0.789 |     0.431 |           |
             |     0.354 |     0.238 |           |
-------------|-----------|-----------|-----------|
         yes |       189 |       628 |       817 |
             |    85.744 |    69.653 |           |
             |     0.231 |     0.769 |     0.409 |
             |     0.211 |     0.569 |           |
             |     0.095 |     0.314 |           |
-------------|-----------|-----------|-----------|
Column Total |       896 |      1103 |      1999 |
             |     0.448 |     0.552 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 628 вчасно було доставлено, а в 707 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 707 з 1182 реальних невчасних доставок,

тобто 59,8 %. Цей тип помилок є потенційно дуже дорогим, оскільки при кожній затримці доставки компанія втрачає клієнтів.

Штучні нейронні мережі (ANN). Для створення базової моделі ANN зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *neuralnet*:

```
> model_tr_ANN=neuralnet(Reached.on.Time_Y.N ~ .,data=delivery_train, hidden=2,act.fct = "logistic",
+                        linear.output = FALSE)
```

Параметр hidden являє собою один прихований шар з 2 нейронами відповідно, а act.fct = "logistic" використовується для згладжування результату.

Візуалізація моделі штучної нейронної мережі виконується за допомогою команди *plot.*

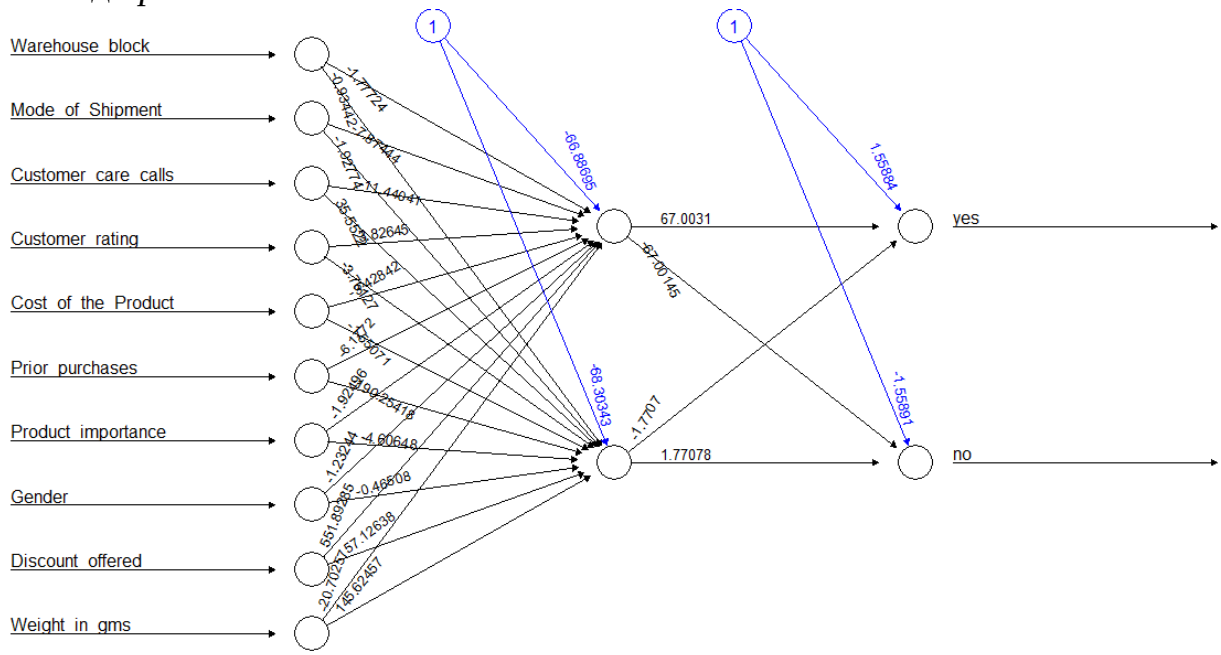


Рисунок 4.6 – Візуалізація результатів моделі ANN для класифікації

Щоб з'ясувати, чи добре навчена модель, необхідно звірити її з даними навчання. Результат отримуємо як таблиці.

```
> pred <- predict(model_tr_ANN, delivery_train)
> table(delivery_train $Reached.on.Time_Y.N, pred[, 1] > 0.5)

      FALSE TRUE
  no   2855 2526
  yes  3506  113
> (3506+2526)/9000
[1] 0.6702222
```

Результат показує, що помилка становить 6032 із 9000, або 67 %. Таким чином, точність класифікації на тренувальних даних – 33 %, що є дуже низьким

показником та можна припущення, що використання нейронних мереж у задачах класифікації не є ефективним.

З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 2855 рази, у той же час неправильно класифіковані 2526 разів. Товари, які були доставлені вчасно правильно передбачені 113 разів, неправильно – 3506 рази. У результаті 2526 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 3506 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Оцінимо модель за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_ANN<-predict(model_tr_ANN,delivery_test)
> delivery_test$testPrediction_ANN <- testPrediction_ANN
> test_ANN<-table(testPrediction_ANN[, 1] > 0.5, delivery_test $Reached.on.Time_Y.N)
> test_ANN

        no yes
  FALSE 621 789
  TRUE  561  28
> (561+789)/1999
[1] 0.6753377
```

Результат показує, що помилка становить 1350 зі 1999, або 67,5 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 32,5 %. Отже, можна зробити висновок, виходячи з оцінки точності класифікації на навчальних та тестових даних, що моделі на нейронних мережах для вирішення задач класифікації не є ефективними.

Вектор *testPrediction_ANN* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_ANN
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))


   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |     FALSE |      TRUE | Row Total |
-------------|-----------|-----------|-----------|
          no |       623 |       559 |      1182 |
             |     0.312 |     0.280 |           |
-------------|-----------|-----------|-----------|
         yes |       797 |        20 |       817 |
             |     0.399 |     0.010 |           |
-------------|-----------|-----------|-----------|
Column Total |      1420 |       579 |      1999 |
-------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для логістичної регресії.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_ANN


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |     FALSE |      TRUE | Row Total |
-------------|-----------|-----------|-----------|
          no |       623 |       559 |      1182 |
             |    55.896 |   137.086 |           |
             |     0.527 |     0.473 |     0.591 |
             |     0.439 |     0.965 |           |
             |     0.312 |     0.280 |           |
-------------|-----------|-----------|-----------|
         yes |       797 |        20 |       817 |
             |    80.868 |   198.330 |           |
             |     0.976 |     0.024 |     0.409 |
             |     0.561 |     0.035 |           |
             |     0.399 |     0.010 |           |
-------------|-----------|-----------|-----------|
Column Total |      1420 |       579 |      1999 |
             |     0.710 |     0.290 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 20 вчасно було доставлено, а в 623 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 623 з 1182 реальних невчасних доставок, тобто 52,7 %.

Випадковий ліс (RF). Для створення базової моделі RF зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *randomForest:*

```
> model_tr_RF<-randomForest(Reached.on.Time_Y.N~.,data=delivery_train)
```

Після налаштування моделі випадкового лісу, на консоль виводиться змінна модель для оцінки частоти помилок.

```
> model_tr_RF

Call:
 randomForest(formula = Reached.on.Time_Y.N ~ ., data = delivery_train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 34.41%
Confusion matrix:
      no  yes class.error
no  3399 1982   0.3683330
yes 1115 2504   0.3080962
> (1115+1982)/9000
[1] 0.3441111
> summary(model_tr_RF)
                Length Class  Mode
call               3  -none- call
type               1  -none- character
predicted       9000  factor numeric
err.rate        1500  -none- numeric
confusion          6  -none- numeric
votes          18000  matrix numeric
oob.times       9000  -none- numeric
classes            2  -none- character
importance        10  -none- numeric
importanceSD       0  -none- NULL
localImportance    0  -none- NULL
proximity          0  -none- NULL
ntree              1  -none- numeric
mtry               1  -none- numeric
forest            14  -none- list
y               9000  factor numeric
test               0  -none- NULL
inbag              0  -none- NULL
terms              3  terms  call
```

Вихідні дані показують, що випадковий ліс складався з 500 дерев і кожне розбиття входило по 3 змінні. Викликає стурбованість низька продуктивність – судячи з матриці невідповідностей, частота помилок на тренувальних даних становить 34,4% (тобто точність дорівнює 65,6%). Однак ця матриця невідповідностей не показує емпіричного ризику, а відображає частоту помилок out-of-bag (OOB) (OOB estimate of error rate), яка на відміну від емпіричного ризику є об'єктивною оцінкою помилки для набору.

Імовірність включення будь-якого спостереження у вибірку bootstrap становить 63,2%. Це означає, що обчисленні частоти помилок OOB за кожен із 9000

спостережень тренувального набору даних проголосувало в середньому 36,8% із 500 дерев випадкового лісу.

Крім того, функція виводить частоту помилок кожного класу (див. матрицю невідповідностей):

- Клас «no» має рівень помилок, що дорівнює 36,8%.

- Клас «yes» має рівень помилок, що дорівнює 30,8%.

Алгоритм випадкового лісу дозволяє побудувати графік помилок для дерев, як показано на рисунку 4.7. На графіку по осі X показано кількість дерев, а на осі Y - рівень помилок. Оскільки в лісі 500 дерев (значення параметра за замовчуванням), обмеження по осі X становить 500. Три різнокольорові криві показують помилку для кожного з 2х класів та загальну частоту помилок у порядку зменшення.

- Клас «no», червона крива (найвищий рівень помилок)

- Загальний рівень помилок, чорна крива (близько 23,1%)

- Клас «yes», зелена крива (найнижчий рівень помилок)

Для створення графіка використано функцію *plot*:

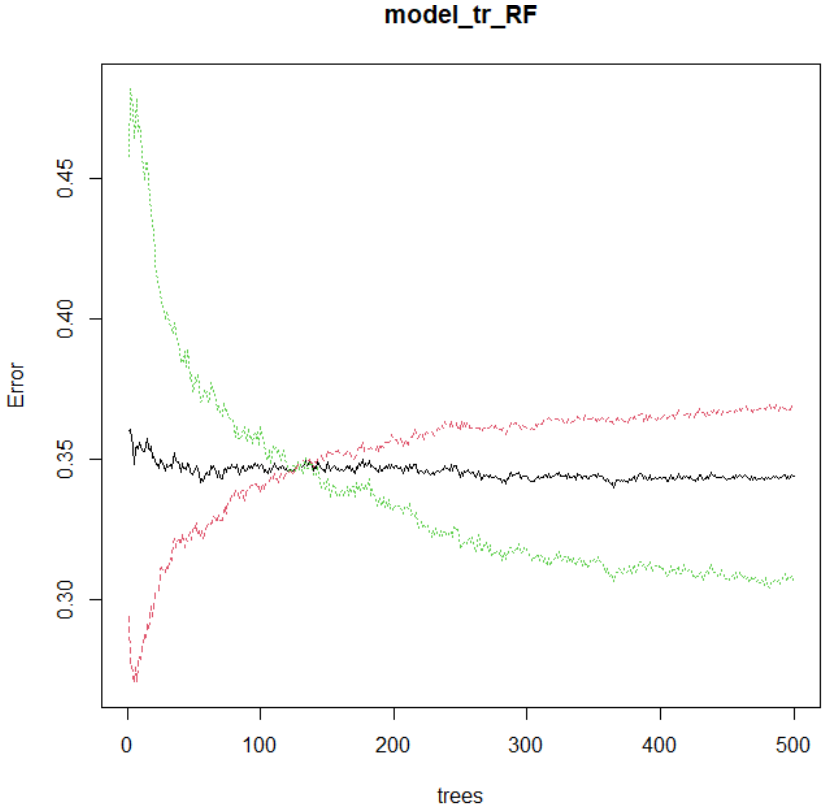


Рисунок 4.7 – Графік помилок для класифікаційної моделі на RF

Оцінимо модель за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_rF<-predict(model_tr_RF,newdata = delivery_test)
> delivery_test$testPrediction_rF <- testPrediction_rF
> table(testPrediction_rF, delivery_test$Reached.on.Time_Y.N)

testPrediction_rF  no yes
               no  738 225
               yes 444 592
> (444+225)/1999
[1] 0.3346673
```

Результат показує, що помилка становить 669 зі 1999, або 33,4 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 66,6 %.

Вектор *testPrediction_rF* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_rF,prop.chisq =
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))


   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       738 |       444 |      1182 |
             |     0.369 |     0.222 |           |
-------------|-----------|-----------|-----------|
         yes |       225 |       592 |       817 |
             |     0.113 |     0.296 |           |
-------------|-----------|-----------|-----------|
Column Total |       963 |      1036 |      1999 |
-------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для логістичної регресії.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_rF
```

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       738 |       444 |      1182 |
             |    49.911 |    46.394 |           |
             |     0.624 |     0.376 |     0.591 |
             |     0.766 |     0.429 |           |
             |     0.369 |     0.222 |           |
-------------|-----------|-----------|-----------|
         yes |       225 |       592 |       817 |
             |    72.209 |    67.120 |           |
             |     0.275 |     0.725 |     0.409 |
             |     0.234 |     0.571 |           |
             |     0.113 |     0.296 |           |
-------------|-----------|-----------|-----------|
Column Total |       963 |      1036 |      1999 |
             |     0.482 |     0.518 |           |
-------------|-----------|-----------|-----------|
```

Модель випадкового лісу правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 592 вчасно було доставлено, а в 738 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 738 з 1182 реальних невчасних доставок, тобто 37 %.

Метод найближчого сусіда (KNN). Для створення базової моделі KNN зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *class*:

```
> train.delivery_labels <- dat[train_sample,1:9]
> test.delivery_labels <-dat[-train_sample,1:9]
> testPrediction_KNN <- knn(train=train.delivery_labels, test=test.delivery_labels, cl=delivery_train$Reached.on.Time_Y.N, k=2)
```

Параметр *cl* – це фрейм істинних класифікацій навчальної множини, *k* – кількість вказаних сусідів. *Train_delivery_labels* – окремий фрейм даних для змінної «Reached.on.Time_Y.N», щоб остаточний результат можна було порівняти з фактичним значенням.

Щоб побачити інформацію про створену модель, скористуємось командою *summary*:

```
> summary(testPrediction_KNN)
  no  yes
1193  806
```

Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> table(testPrediction_KNN,delivery_test$Reached.on.Time_Y.N)

testPrediction_KNN  no yes
                no  790 403
                yes 392 414
> (392+403)/1999
[1] 0.3976988
```

Результат показує, що помилка становить 795 зі 1999, або 39,7 %. Це практично так само, як і на навчальних даних. Отже, точність класифікації на тестових даних – 60,3 %. З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 790 рази, у той же час неправильно класифіковані 403 разів. Товари, які були доставлені вчасно правильно передбачені 414 разів, неправильно – 392 рази. У результаті 403 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 392 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Вектор *testPrediction_KNN* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_KNN,prop.chisq =
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))

   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  1999

             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       790 |       392 |      1182 |
             |     0.395 |     0.196 |           |
-------------|-----------|-----------|-----------|
         yes |       403 |       414 |       817 |
             |     0.202 |     0.207 |           |
-------------|-----------|-----------|-----------|
Column Total |      1193 |       806 |      1999 |
-------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для метода найближчого сусіда.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_KNN,
```

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  1999

             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       790 |       392 |      1182 |
             |    10.142 |    15.012 |           |
             |     0.668 |     0.332 |     0.591 |
             |     0.662 |     0.486 |           |
             |     0.395 |     0.196 |           |
-------------|-----------|-----------|-----------|
         yes |       403 |       414 |       817 |
             |    14.673 |    21.719 |           |
             |     0.493 |     0.507 |     0.409 |
             |     0.338 |     0.514 |           |
             |     0.202 |     0.207 |           |
-------------|-----------|-----------|-----------|
Column Total |      1193 |       806 |      1999 |
             |     0.597 |     0.403 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 414 вчасно було доставлено, а в 790 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 790 з 1182 реальних невчасних доставок, тобто 66,8%.

4.2 Результати ансамблю моделей дворівневої класифікації

Для досягнення найкращої якості класифікації моделі часто вдаються до ансамблевих методів багаторівневої класифікації. Їх ідея полягає у використанні декількох моделей (алгоритмів класифікації) на різних шарах. Одна з переваг даного методу полягає в тому, що забезпечується гнучкість вибору вирішальних функцій, що знаходять "приховану залежність" між об'єктами навчальної вибірки та відповідними значеннями їх класів (через всілякі комбінації вирішальних функцій базових класифікаторів, що входять до складу ансамблевого).

Одними з найвідоміших ансамблевих алгоритмів є *stacking, bagging* та *boosting.*

Більшість моделей мають непогану якість роботи, але результат все ще потребує покращення, оскільки компанія прагне покращити сервіс надавання послуг з продажу та доставки товарів. Тому для покращення результату побудуємо дворівневу ансамблеву модель. У якості вхідних даних обираємо тестові оцінки якості базових моделей та додаємо їх до нашого тестового набору.

```
> delivery_test$testPrediction_tr <- testPrediction_tr
> delivery_test$testPrediction_NB <- testPrediction_NB
> delivery_test$testPrediction_QDA <- testPrediction_QDA$class
> delivery_test$testPrediction_LR <- testPrediction_LR
> delivery_test$testPrediction_LDA <- testPrediction_LDA$class
> delivery_test$testPrediction_svm <- testPrediction_svm
> delivery_test$testPrediction_ANN <- testPrediction_ANN
> delivery_test$testPrediction_rF <- testPrediction_rF
> delivery_test$testPrediction_KNN<-testPrediction_KNN
```

На *першому шарі* скористуємось методом стекінгу на основі лінійної регресійної моделі. Модельний стекінг є ефективним методом ансамблювання. Прогнози, які сформовані за допомогою базових моделей, використовуються як вхідні дані для навчання на першому шарі. Наприклад, лінійна регресія оцінює подані на вхід моделі шляхом мінімізації помилок з найменшими квадратами.

В якості базових моделей першого шару обрано:

- дерева рішень (DecisionTree),
- наївний Баєсів класифікатор (NB),
- квадратичний дискримінантний аналіз (QDA),
- логістична регресія (LR),
- метод опорних векторів (SVM),
- модель випадкового лісу (RF).

Перший шар: Stacking. Для створення ансамблевої моделі першого шару базових моделей зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *caret*. Кілька обраних базових моделей об’єднуються за допомогою пакета *caretEnsemble*. Потім, враховуючи список вхідних моделей, функція *caretStack()* використовується для визначення моделі вищого порядку, щоб якнайкраще об’єднати результати підмоделей.

Функція *trainControl()* визначає контрольні параметри, які будуть використовуватися під час моделювання. Список *predictors* містить перелік базових моделей для тренувального набору даних з *delivery_train*, функція *caretList()* навчає модель та об'єднаємо прогнози класифікаторів за допомогою функції *caretStack()*.

```
> control <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE, classProbs=TRUE)
> predictors<-c('testPrediction_svm','testPrediction_tr','testPrediction_rF',
+ 'testPrediction_NB','testPrediction_QDA','testPrediction_LR')
> models<-caretList(delivery_test[,predictors],delivery_test$Reached.on.Time_Y.
+ ,trControl=control,methodList=c("glm"))
> stackControl <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE, classProbs='
> stack.glm <- caretStack(models, method="glm", trControl=stackControl)
```

Коли ми об'єднуємо прогнози різних моделей за допомогою стекінгу, бажано, щоб прогнози, зроблені підмоделями, мали низьку кореляцію. Це означає, що моделі є вправними та дозволяє новому класифікатору з'ясувати, як отримати найкраще від кожної моделі для покращеного результату.

```
> summary(models)
    Length Class Mode
glm 21     train list
```

Маючи список типу *caretList*, функція *caretStack()* може бути використана для визначення моделі вищого порядку, щоб дізнатися, як найкраще об'єднати прогнози підмоделей разом.

Основні відомості про отриману ансамблеву модель першого шару на основі стекінгу:

```
> stack.glm
A glm ensemble of 1 base models: glm

Ensemble results:
Generalized Linear Model

5997 samples
   1 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 5397, 5398, 5398, 5396, 5398, 5397,
Resampling results:

  Accuracy   Kappa
  0.6936269  0.4235448
```

Основні відомості ансамблю моделей першого шару: логістична регресія, кількість предикторів: 1, два класи результуючої змінної: "no" та "yes", точність тренувальної моделі 69,3 % та каппа-статистика 0,42. Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб

оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

Результат показує, що помилка становить 1399 зі 1999, або 30 %. Отже, точність класифікації на тестових даних – 70 %. З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 607 рази, у той же час неправильно класифіковані 25 разів. Товари, які були доставлені вчасно правильно передбачені 792 разів, неправильно – 575 рази. У результаті 25 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 575 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

```
> testPrediction_stack<-predict(stack.glm,newdata= delivery_test)
> delivery_test$testPrediction_stack<-testPrediction_stack
> table(testPrediction_stack, delivery_test$Reached.on.Time_Y.N)

testPrediction_stack  no yes
                  no  607  25
                  yes 575 792
> (575+25)/1999
[1] 0.3001501
```

Вектор *testPrediction_stack* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_stack,
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))


   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       607 |       575 |      1182 |
             |     0.304 |     0.288 |           |
-------------|-----------|-----------|-----------|
         yes |        25 |       792 |       817 |
             |     0.013 |     0.396 |           |
-------------|-----------|-----------|-----------|
Column Total |       632 |      1367 |      1999 |
-------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для ансамблю моделі першого рівня.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_stack,

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1999


             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       607 |       575 |      1182 |
             |   145.651 |    67.338 |           |
             |     0.514 |     0.486 |     0.591 |
             |     0.960 |     0.421 |           |
             |     0.304 |     0.288 |           |
-------------|-----------|-----------|-----------|
         yes |        25 |       792 |       817 |
             |   210.721 |    97.422 |           |
             |     0.031 |     0.969 |     0.409 |
             |     0.040 |     0.579 |           |
             |     0.013 |     0.396 |           |
-------------|-----------|-----------|-----------|
Column Total |       632 |      1367 |      1999 |
             |     0.316 |     0.684 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 792 вчасно було доставлено, а в 607 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 607 з 1182 реальних невчасних доставок, тобто 51,4%.

Отже, в результаті об’єднання деяких базових моделей в ансамбль моделі за допомогою стекінгу, нам вдалось підвищити точність моделі на першому рівні до 70 %.

На *другому шарі* скористуємось методом беггінгу на основі алгоритму Bagged CART. Алгоритм створює N регресійних дерев, використовуючи M початкових навчальних наборів і усереднює отримані прогнози. Ці дерева вирощуються глибоко та не обрізаються. Кожне окреме дерево має високу дисперсію, але низьке похибку. Усереднення N дерев зменшує дисперсію. Прогнозованими значеннями для спостережень є мода (класифікація) або середнє значення (регресія) дерев. Одним із недоліків Bagged Trees є те, що невелика

кількість додаткових навчальних спостережень може різко змінити ефективність передбачення вивченого дерева.

В якості базових моделей другого шару обрано:

- модель першого рівня (Stacking LR),
- штучних нейронних мереж (ANN),
- лінійний дискримінантний аналіз (LDA),
- метод найближчого сусіда (KNN).

Другий шар: Bagging. Для створення ансамблевої моделі другого шару базових моделей зі змінною Reached.on.Time_Y.N як відгук та всіх інших змінних як предикторів використовують пакет *caret*. Метод *treebag* найкраще працює з алгоритмами, які мають високу дисперсію, наприклад, дерева рішень. Параметр *controlBag* визначає параметри, які використовуються для керування процесом навчання моделі та функція *train()*, що навчає алгоритм. Аргумент *method="treebag"* визначає алгоритм навчання.

```
> predictorsBag<-c('testPrediction_stack','testPrediction_ANN','testPrediction_KNN','testPrediction_LDA')
> controlBag <- trainControl(method="repeatedcv", number=10, repeats=3)
> bagCART_model<-train(delivery_test[,predictorsBag],delivery_test$Reached.on.Time_Y.,method='treebag',trControl=controlBag)
```

Основні відомості про отриману ансамблеву модель другого шару на основі беггінгу:

```
> bagCART_model
Bagged CART

1999 samples
   4 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1799, 1799, 1800, 1799, 1798, 1800, ...
Resampling results:

  Accuracy   Kappa
  0.6638178  0.3307129
```

Основні відомості ансамблю моделей другого шару: метод Bagged CART, кількість предикторів: 4, два класи результуючої змінної: “no” та “yes”, точність тренувальної моделі 66,3 % та каппа-статистика 0,33. Попередньо вважаємо, що побудовано непогану модель. Оцінимо її за допомогою тестових даних. Щоб оцінити, як модель працює з тестовими даними, переглядаємо таблицю та обчислюємо помилку.

```
> testPrediction_bag<-predict(bagCART_model,newdata= delivery_test)
> table(testPrediction_bag, delivery_test $Reached.on.Time_Y.N)

testPrediction_bag  no yes
               no  959   9
               yes 223 808
> (223+9)/1999
[1] 0.116058
```

Результат показує, що помилка становить 232 зі 1999, або 11 %. Отже, точність класифікації на тестових даних – 88 %. З наведеної таблиці видно, що невчасно доставлені товари правильно передбачені 959 рази, у той же час неправильно класифіковані 9 разів. Товари, які були доставлені вчасно правильно передбачені 808 разів, неправильно – 223 рази. У результаті 9 реальних значень “no” були помилково класифіковані як “yes” (хибно-позитивні спрацьовування), а 223 значення “yes” були помилково класифіковані як “no” (хибно-негативні спрацьовування).

Вектор *testPrediction_bag* містить прогнозовані значення класу, які можна порівнювати з реальними значеннями класу, використовуючи функцію *CrossTable()*. Наведемо розширену матрицю невідповідностей.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_bag,prop.chisq =
+ FALSE,prop.c = FALSE,prop.r = FALSE,dnn=c('actual','predicted'))
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  1999

              | predicted
       actual |        no |       yes | Row Total |
--------------|-----------|-----------|-----------|
           no |       959 |       223 |      1182 |
              |     0.480 |     0.112 |           |
--------------|-----------|-----------|-----------|
          yes |         9 |       808 |       817 |
              |     0.005 |     0.404 |           |
--------------|-----------|-----------|-----------|
 Column Total |       968 |      1031 |      1999 |
--------------|-----------|-----------|-----------|
```

Також, наведемо матрицю невідповідностей, крос-таблицю для ансамблю моделі другого рівня.

```
> CrossTable(delivery_test$Reached.on.Time_Y.N,testPrediction_bag,dnn=c('actual','predicted'))

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  1999

             | predicted
      actual |        no |       yes | Row Total |
-------------|-----------|-----------|-----------|
          no |       959 |       223 |      1182 |
             |   261.157 |   245.199 |           |
             |     0.811 |     0.189 |     0.591 |
             |     0.991 |     0.216 |           |
             |     0.480 |     0.112 |           |
-------------|-----------|-----------|-----------|
         yes |         9 |       808 |       817 |
             |   377.831 |   354.743 |           |
             |     0.011 |     0.989 |     0.409 |
             |     0.009 |     0.784 |           |
             |     0.005 |     0.404 |           |
-------------|-----------|-----------|-----------|
Column Total |       968 |      1031 |      1999 |
             |     0.484 |     0.516 |           |
-------------|-----------|-----------|-----------|
```

Ця модель правильно визначила, що зі 1999 оброблених доставок товару, що входять до тестового набору даних, 808 вчасно було доставлено, а в 959 випадках відправлення доставляється із затримкою. Слід звернути увагу, що у тестових даних модель правильно спрогнозувала 959 з 1182 реальних невчасних доставок, тобто 81,1%.

Отже, на другому етапі в результаті об’єднання деяких базових моделей та моделі стекінгу в ансамбль моделі за допомогою беггінгу, нам вдалось підвищити точність моделі на другому рівні до 88 %, що є дуже гарним результатом.

4.3 Оцінка ефективності розроблених моделей класифікації

Досі точність класифікатора вимірювалася як кількість правильних прогнозів, поділена на загальну кількість прогнозів. Таким чином отримували частку випадків, у яких алгоритм навчання мав рацію. Однак було б розумно зібрати додаткову інформацію щодо ефективності класифікатора, перш ніж використовувати його на практиці для прогнозування. Оцінюючи класифікатора необхідно пам'ятати про проблему дисбалансу класів, що виникає, коли переважна

більшість записів у наборі даних відносяться до одного класу. Ця проблема є і в досліджуваному наборі даних.

Є багато способів виміряти ефективність класифікатора, проте найкращий показник – той, який визначає, чи є класифікатор успішним, коли застосовується за прямим призначенням. Важливо визначити показники ефективності, які показуватимуть корисність, а чи не просто точність. Альтернативні показники ефективності отримаємо із матриці невідповідностей.

Матриця невідповідностей – це таблиця, яка класифікує прогнози залежно від цього, чи відповідають вони фактичним значенням. Один із вимірів таблиці – це можливі категорії прогнозованих значень, а друге – ті самі категорії для реальних значень.

Якщо прогнозоване значення збігається з реальним, класифікація правильна. Правильні прогнози розміщуються у матриці невідповідностей щодо діагоналі. Інші комірки матриці відповідають випадкам, коли прогнозоване значення відрізняється від реального. Показники ефективності для моделей класифікації засновані на кількості прогнозів, що потрапляють на діагональ та за її межі.

Розглянемо найбільш поширені показники ефективності, що враховують здатність моделі відрізняти один клас від інших: *точність прогнозування, каппа-статистика, чутливість та специфічність, точність та повнота, F- міра та площа під ROC - кривою.*

Точність прогнозування

В R є можливість для автоматичного розрахунку точності на основі векторів прогнозованих і реальних значень. Для обчислення точності прогнозування застосовуються функція *Accuracy* пакета *caret*.

Базові моделі:

- Дерева рішень (Decision Tree)

```
> Accuracy(testPrediction_tr,delivery_test$Reached.on.Time_Y.N)
[1] 0.7013507
```

- Наївний Баєсів класифікатор (NB)

```
> Accuracy(testPrediction_NB,delivery_test$Reached.on.Time_Y.N)
[1] 0.6738369
```

- Лінійний дискримінантний аналіз (LDA)

```
> Accuracy(testPrediction_LDA$class,delivery_test$Reached.on.Time_Y.N)
[1] 0.6433217
```

- Квадратичний дискримінантний аналіз (QDA)

```
> Accuracy(testPrediction_QDA$class,delivery_test$Reached.on.Time_Y.N)
[1] 0.6668334
```

- Логістична регресія (LR)

```
[1] "Accuracy:"
[1] 0.6453227
```

- Метод опорних векторів (SVM)

```
> Accuracy(testPrediction_svm,delivery_test$Reached.on.Time_Y.N)
[1] 0.6678339
```

- Метод найближчого сусіда (KNN)

```
> Accuracy(testPrediction_KNN,delivery_test$Reached.on.Time_Y.N)
[1] 0.6003002
```

- Штучні нейронні мережі (ANN)

```
[1] "Accuracy:"
[1] 0.3256628
```

- Модель випадкового лісу (RF)

```
> Accuracy(testPrediction_rF,delivery_test$Reached.on.Time_Y.N)
[1] 0.6688344
```

Дворівневий ансамбль моделі:

- Стекінг (Stacking LR)

```
> Accuracy(testPrediction_stack,delivery_test$Reached.on.Time_Y.N)
[1] 0.6998499
```

- Беггінг (Bagged CART)

```
> Accuracy(testPrediction_bag,delivery_test$Reached.on.Time_Y.N)
[1] 0.883942
```

Каппа-статистика

У R є функції автоматичного розрахунку коефіцієнтів каппа. У функції Kappa() з пакету Visualizing Categorical Data (vcd) використовується матриця невідповідностей прогнозованих та реальних значень.

Базові моделі:

- Дерева рішень (Decision Tree)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_tr))
             value    ASE    z   Pr(>|z|)
Unweighted 0.4423 0.0162 27.3 3.955e-164
Weighted   0.4423 0.0162 27.3 3.955e-164
```

- Наївний Баєсів класифікатор (NB)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_NB))
             value    ASE     z  Pr(>|z|)
Unweighted 0.3643 0.0195 18.68 6.962e-78
Weighted   0.3643 0.0195 18.68 6.962e-78
```

- Лінійний дискримінантний аналіз (LDA)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_LDA$class))
             value      ASE     z  Pr(>|z|)
Unweighted 0.2674 0.02175 12.3 9.216e-35
Weighted   0.2674 0.02175 12.3 9.216e-35
```

- Квадратичний дискримінантний аналіз (QDA)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_QDA$class))
             value    ASE     z    Pr(>|z|)
Unweighted 0.3846 0.0158 24.34 7.821e-131
Weighted   0.3846 0.0158 24.34 7.821e-131
```

- Логістична регресія (LR)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_LR))
             value      ASE     z  Pr(>|z|)
Unweighted 0.2691 0.02178 12.36 4.458e-35
Weighted   0.2691 0.02178 12.36 4.458e-35
```

- Метод опорних векторів (SVM)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_svm))
             value      ASE     z  Pr(>|z|)
Unweighted 0.348 0.01995 17.44 3.912e-68
Weighted   0.348 0.01995 17.44 3.912e-68
```

- Метод найближчого сусіда (KNN)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_KNN))
             value      ASE    z  Pr(>|z|)
Unweighted 0.1694 0.02226 7.61 2.747e-14
Weighted   0.1694 0.02226 7.61 2.747e-14
```

- Штучні нейронні мережі (ANN)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_ANN[, 1] > 0.5))
               value      ASE       z    Pr(>|z|)
Unweighted -0.4574 0.01548 -29.54 8.386e-192
Weighted   -0.4574 0.01548 -29.54 8.386e-192
```

- Модель випадкового лісу (RF)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_rF))
             value    ASE     z  Pr(>|z|)
Unweighted 0.3412 0.0205 16.64 3.391e-62
Weighted   0.3412 0.0205 16.64 3.391e-62
```

Дворівневий ансамбль моделі:

- Стекінг (Stacking LR)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_stack))
             value      ASE     z    Pr(>|z|)
Unweighted 0.4375 0.01652 26.49 1.305e-154
Weighted   0.4375 0.01652 26.49 1.305e-154
```

- Беггінг (Bagged CART)

```
> Kappa(table(delivery_test$Reached.on.Time_Y.N,testPrediction_bag))
             value      ASE     z Pr(>|z|)
Unweighted 0.7692 0.01392 55.26        0
Weighted   0.7692 0.01392 55.26        0
```

Чутливість та специфічність

В R є можливість для автоматичного розрахунку коефіцієнтів чутливості і специфічності безпосередньо на основі векторів прогнозованих і реальних значень. Для обчислення коефіцієнтів ефективності застосовуються функції *specificity* та *sensitivity* пакета *caret*.

Чутливість базових моделей:

- Дерева рішень (Decision Tree)

```
> sensitivity(testPrediction_tr,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.5059222
```

- Наївний Баєсів класифікатор (NB)

```
> sensitivity(testPrediction_NB,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.5862944
```

- Лінійний дискримінантний аналіз (LDA)

```
> sensitivity(testPrediction_LDA$class,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.6818951
```

- Квадратичний дискримінантний аналіз (QDA)

```
> sensitivity(testPrediction_QDA$class,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.4492386
```

- Логістична регресія (LR)

```
[1] "Sensitivity:"
[1] 0.5644391
```

- Метод опорних векторів (SVM)

```
> sensitivity(testPrediction_svm,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.5981387
```

- Метод найближчого сусіда (KNN)

```
> sensitivity(testPrediction_KNN,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.6717428
```

- Штучні нейронні мережі (ANN)

```
[1] "Sensitivity:"
[1] 0.05076142
```

- Модель випадкового лісу (RF)

```
> sensitivity(testPrediction_rF,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.6302876
```

Чутливість дворівневого ансамблю моделі:

- Стекінг (Stacking LR)

```
> sensitivity(testPrediction_stack,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.5135364
```

- Беггінг (Bagged CART)

```
> sensitivity(testPrediction_bag,delivery_test$Reached.on.Time_Y.N,negative="no")
[1] 0.8113367
```

Специфічність базових моделей:

- Дерева рішень (Decision Tree)

```
> specificity(testPrediction_tr,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.9840881
```

- Наївний Баєсів класифікатор (NB)

```
> specificity(testPrediction_NB,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.8004896
```

- Лінійний дискримінантний аналіз (LDA)

```
> specificity(testPrediction_LDA$class,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.5875153
```

- Квадратичний дискримінантний аналіз (QDA)

```
> specificity(testPrediction_QDA$class,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.9816401
```

- Логістична регресія (LR)

```
[1] "Specificity:"
[1] 0.7037037
```

- Метод опорних векторів (SVM)

```
> specificity(testPrediction_svm,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.7686659
```

- Метод найближчого сусіда (KNN)

```
> specificity(testPrediction_KNN,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.49694
```

- Штучні нейронні мережі (ANN)

```
[1] "Specificity:"
[1] 0.4410511
```

- Модель випадкового лісу (RF)

```
> specificity(testPrediction_rF,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.7246022
```

Специфічність дворівневого ансамблю моделі:

- Стекінг (Stacking LR)

```
> specificity(testPrediction_stack,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.9694002
```

- Беггінг (Bagged CART)

```
> specificity(testPrediction_bag,delivery_test$Reached.on.Time_Y.N,positive="yes")
[1] 0.9889841
```

Точність та повнота

Використовуючи пакет *caret* можна автоматично розрахувати коефіцієнти точності та повноти на основі векторів спрогнозованих та реальних класів. Для обчислення точності використовується функція *precision()* та обчислення повноти застосовується функція *recall()*.

Точність базових моделей:

- Дерева рішень (Decision Tree)

```
> precision(testPrediction_tr,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.9787234
```

- Наївний Баєсів класифікатор (NB)

```
> precision(testPrediction_NB,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.8095794
```

- Лінійний дискримінантний аналіз (LDA)

```
> precision(testPrediction_LDA$class,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.7051619
```

- Квадратичний дискримінантний аналіз (QDA)

```
> precision(testPrediction_QDA$class,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.9725275
```

- Логістична регресія (LR)

```
[1] "Precision:"
[1] 0.5789474
```

- Метод опорних векторів (SVM)

```
> precision(testPrediction_svm,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.7890625
```

- Метод найближчого сусіда (KNN)

```
> precision(testPrediction_KNN,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.6589212
```

- Штучні нейронні мережі (ANN)

```
[1] "Precision:"
[1] 0.03671971
```

- Модель випадкового лісу (RF)

```
> precision(testPrediction_stack,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.960443
```

Точність дворівневого ансамблю моделі:

- Стекінг (Stacking LR)

```
> precision(testPrediction_stack,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.960443
```

- Беггінг (Bagged CART)

```
> precision(testPrediction_bag,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.9816327
```

Повнота базової моделі:

- Дерева рішень (Decision Tree)

```
> recall(testPrediction_tr,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.5059222
```

- Наївний Баєсів класифікатор (NB)

```
 > recall(testPrediction_NB,delivery_test$Reached.on.Time_Y.N,positive="good")
 [1] 0.5862944
```

- Лінійний дискримінантний аналіз (LDA)

```
> recall(testPrediction_LDA$class,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.6818951
```

- Квадратичний дискримінантний аналіз (QDA)

```
> recall(testPrediction_QDA$class,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.4492386
```

- Логістична регресія (LR)

```
[1] "Recall:"
[1] 0.5644391
```

- Метод опорних векторів (SVM)

```
> recall(testPrediction_svm,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.5981387
```

- Метод найближчого сусіда (KNN)

```
> recall(testPrediction_KNN,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.6717428
```

- Штучні нейронні мережі (ANN)

```
[1] "Recall:"
[1] 0.05076142
```

- Модель випадкового лісу (RF)

```
> recall(testPrediction_rF,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.6302876
```

Повнота дворівневого ансамблю моделі:

- Стекінг (Stacking LR)

```
> recall(testPrediction_stack,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.5135364
```

- Беггінг (Bagged CART)

```
> recall(testPrediction_bag,delivery_test$Reached.on.Time_Y.N,positive="good")
[1] 0.8138748
```

F-міра

В R є можливість для автоматичного розрахунку міри ефективності моделі безпосередньо на основі векторів прогнозованих і реальних значень. Для обчислення F-міри застосовуються функція *F1_Score()* пакета *caret*.

Базові моделі:

- Дерева рішень (Decision Tree)

```
> F1_Score(testPrediction_tr,delivery_test$Reached.on.Time_Y.N)
[1] 0.6670385
```

- Наївний Баєсів класифікатор (NB)

```
> F1_Score(testPrediction_NB,delivery_test$Reached.on.Time_Y.N)
[1] 0.6800785
```

- Лінійний дискримінантний аналіз (LDA)

```
> F1_Score(testPrediction_LDA$class,delivery_test$Reached.on.Time_Y.N)
[1] 0.6933333
```

- Квадратичний дискримінантний аналіз (QDA)

```
> F1_Score(testPrediction_QDA$class,delivery_test$Reached.on.Time_Y.N)
[1] 0.6145833
```

- Логістична регресія (LR)

```
[1] "F1-Score:"
[1] 0.5716012
```

- Метод опорних векторів (SVM)

```
> F1_Score(testPrediction_svm,delivery_test$Reached.on.Time_Y.N)
[1] 0.680462
```

- Метод найближчого сусіда (KNN)

```
> F1_Score(testPrediction_KNN,delivery_test$Reached.on.Time_Y.N)
[1] 0.6652702
```

- Штучні нейронні мережі (ANN)

```
[1] "F1-Score:"
[1] 0.04261364
```

- Модель випадкового лісу (RF)

```
> F1_Score(testPrediction_rF,delivery_test$Reached.on.Time_Y.N)
[1] 0.6923792
```

Дворівневий ансамбль моделі:

- Стекінг (Stacking LR)

```
> F1_Score(testPrediction_stack,delivery_test$Reached.on.Time_Y.N)
[1] 0.6692393
```

- Беггінг (Bagged CART)

```
> F1_Score(testPrediction_bag,delivery_test$Reached.on.Time_Y.N)
[1] 0.8899167
```

Площа під ROC – кривою

В R є можливість для автоматичного розрахунку ROC-кривої моделі безпосередньо на основі векторів прогнозованих і реальних значень. Для обчислення AUC ROC застосовуються функції *roc()* та *auc()* пакета pROC.

Базові моделі:

- Дерева рішень (Decision Tree)

```
> roc_object_tr <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_tr))
> plot(roc_object_tr,print.auc=TRUE)
> lines(roc_object_tr, col="red", type='b')
> auc(roc_object_tr)
Area under the curve: 0.745
```

- Наївний Баєсів класифікатор (NB)

```
> roc_object_nb <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_NB))
> plot(roc_object_nb,print.auc=TRUE)
> lines(roc_object_nb, col="red", type='b')
> auc(roc_object_nb)
Area under the curve: 0.6934
```

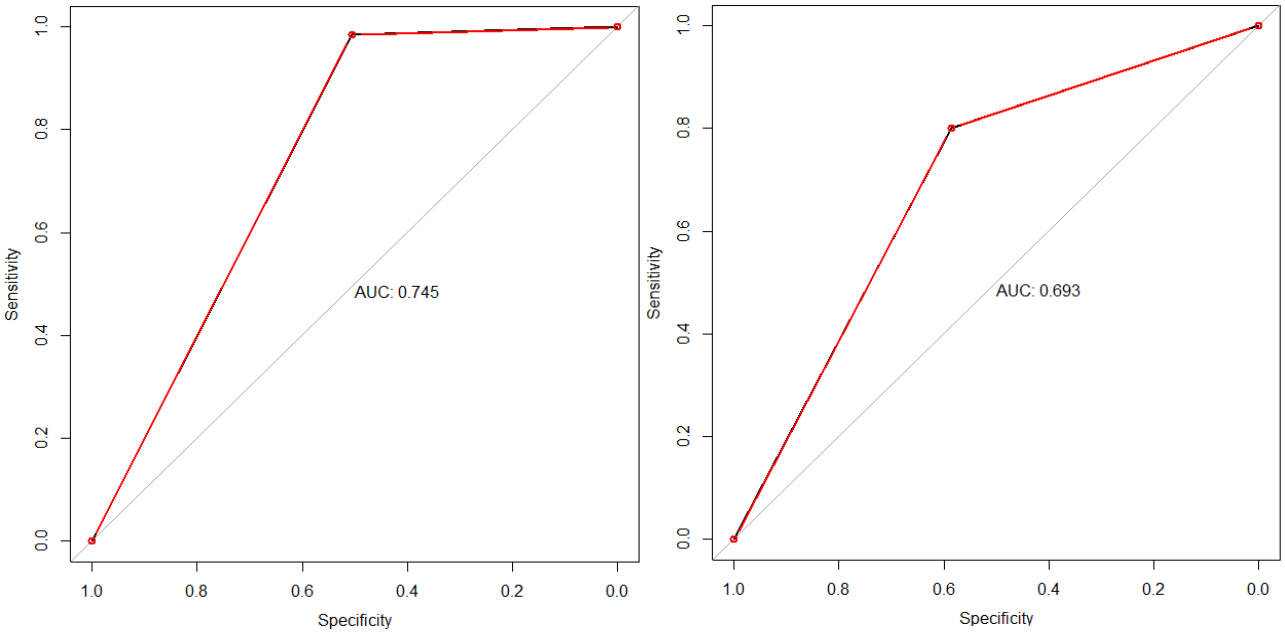


Рисунок 4.8 – ROC-криві моделей дерева рішень та наївного Баєсівського класифікатору

- Лінійний дискримінантний аналіз (LDA)

```
> roc_object_lda <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_LDA$class))
> plot(roc_object_lda,print.auc=TRUE)
> lines(roc_object_lda, col="red", type='b')
> auc(roc_object_lda)
Area under the curve: 0.6347
```

- Квадратичний дискримінантний аналіз (QDA)

```
> roc_object_qda <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_QDA$class))
> plot(roc_object_qda,print.auc=TRUE)
> lines(roc_object_qda, col="red", type='b')
> auc(roc_object_qda)
Area under the curve: 0.7154
```

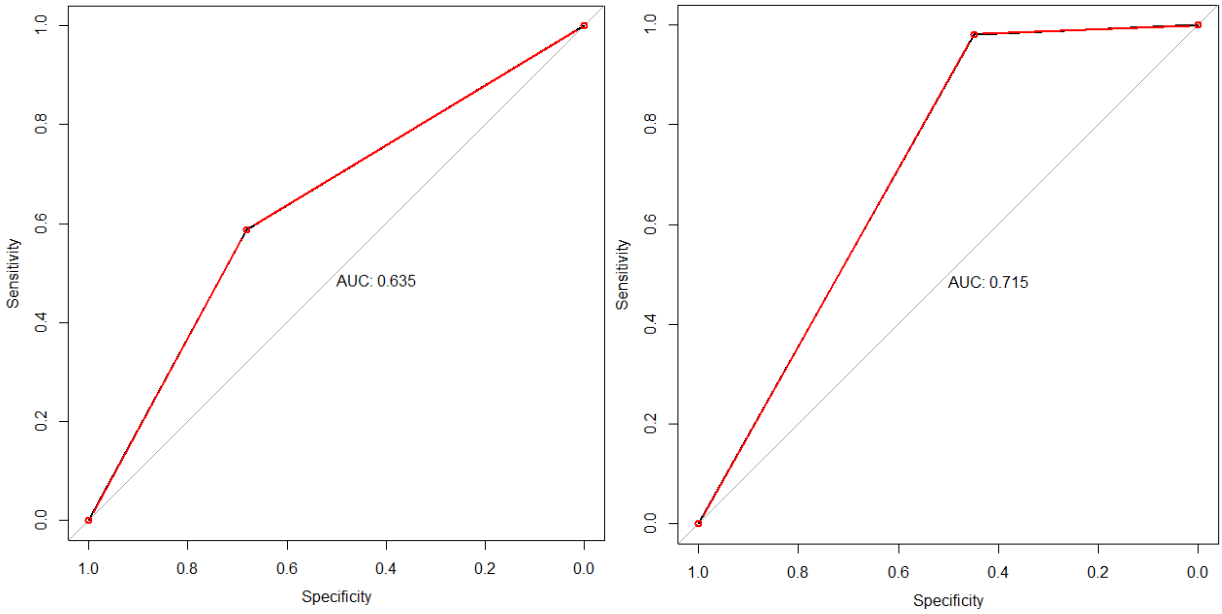


Рисунок 4.9 – ROC-криві моделей лінійного та квадратичного дискримінантного аналізу

- Логістична регресія (LR)

```
> roc_object_lr <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_LR))
> plot(roc_object_lr,print.auc=TRUE)
> lines(roc_object_lr, col="red", type='b')
> auc(roc_object_lr)
Area under the curve: 0.6351
```

- Метод опорних векторів (SVM)

```
> roc_object_svm <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_svm))
> plot(roc_object_svm,print.auc=TRUE)
> lines(roc_object_svm, col="red", type='b')
> auc(roc_object_svm)
Area under the curve: 0.6834
```

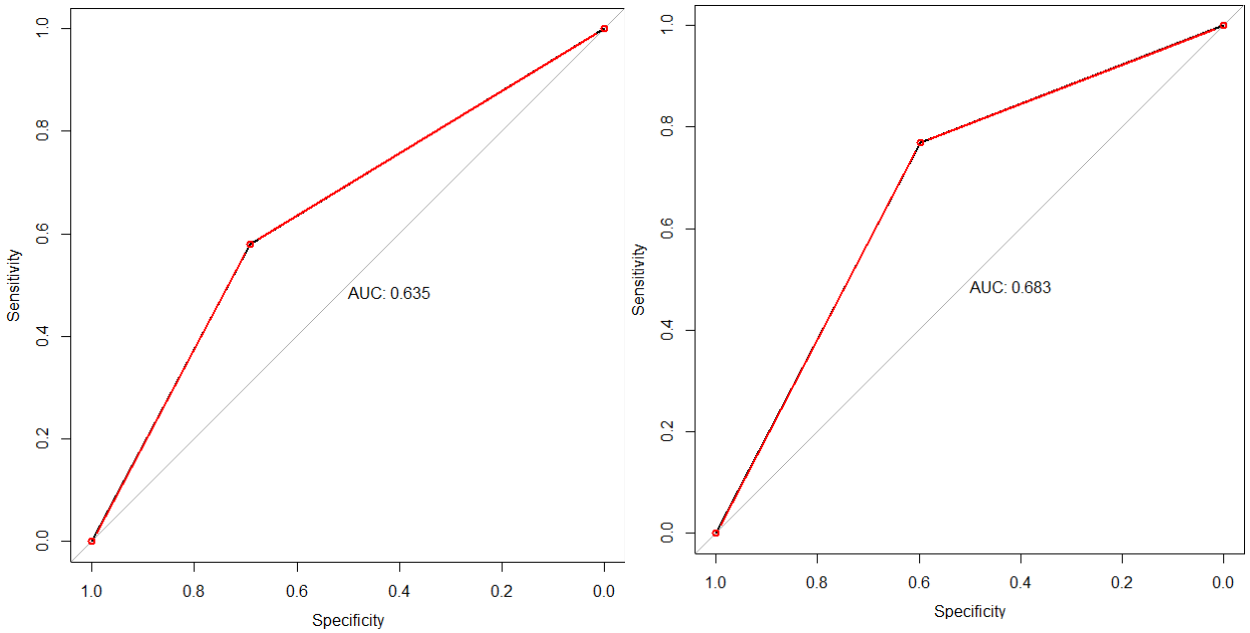


Рисунок 4.10 – ROC-криві моделей логістичної регресії та метода опорних векторів

- Метод найближчого сусіда (KNN)

```
> roc_object_ann <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_ANN[,1] > 0.5))
> plot(roc_object_ann,print.auc=TRUE)
> lines(roc_object_ann, col="red", type='b')
> auc(roc_object_ann)
Area under the curve: 0.2811
```

- Штучні нейронні мережі (ANN)

```
> roc_object_knn <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_KNN))
> plot(roc_object_knn,print.auc=TRUE)
> lines(roc_object_knn, col="red", type='b')
> auc(roc_object_knn)
Area under the curve: 0.5843
```

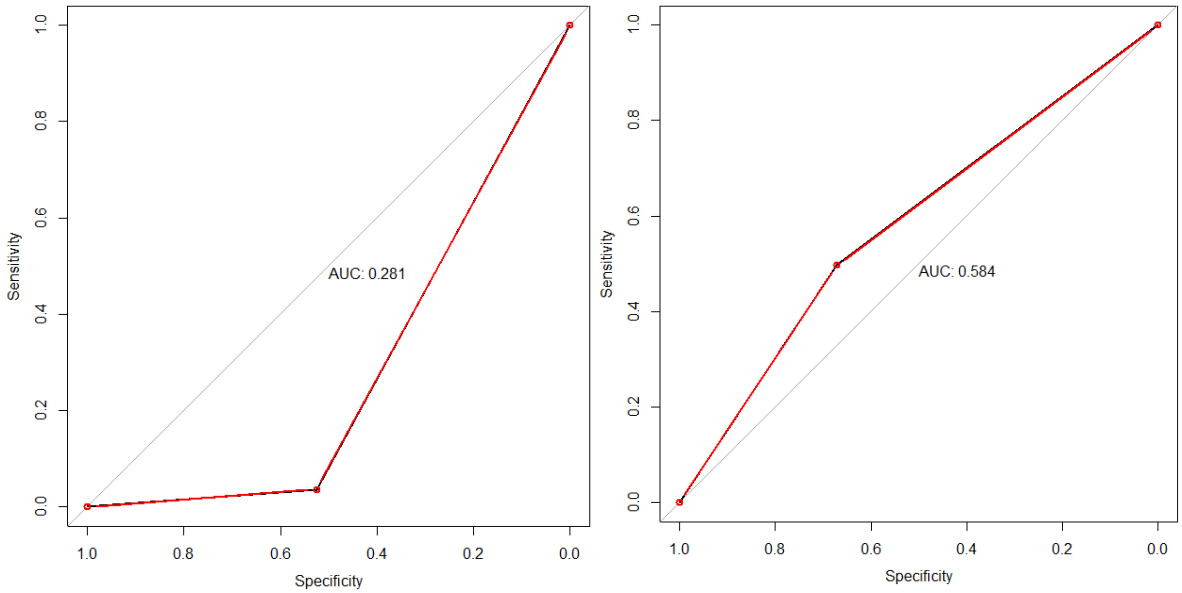


Рисунок 4.11 – ROC-криві моделей методу найближчого сусіда та ANN

Модель випадкового лісу (RF)

```
> roc_object_rF <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_rF))
> plot(roc_object_rF,print.auc=TRUE)
> lines(roc_object_rF, col="red", type='b')
> auc(roc_object_rF)
Area under the curve: 0.6774
```

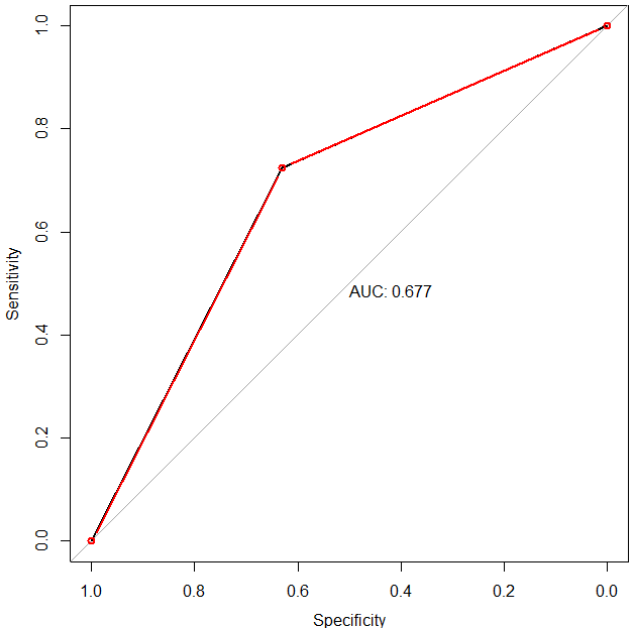


Рисунок 4.12 – ROC-криві моделі випадкового лісу

Дворівневий ансамбль моделі:

- Стекінг (Stacking LR)

```
> roc_object_stack <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_stack))
> plot(roc_object_stack,print.auc=TRUE)
> lines(roc_object_stack, col="red", type='b')
> auc(roc_object_stack)
Area under the curve: 0.7415
```

- Беггінг (Bagged CART)

```
> roc_object_bag <- roc(delivery_test$Reached.on.Time_Y.N, as.numeric(testPrediction_bag))
> plot(roc_object_bag,print.auc=TRUE)
> lines(roc_object_bag, col="red", type='b')
> auc(roc_object_bag)
Area under the curve: 0.8959
```

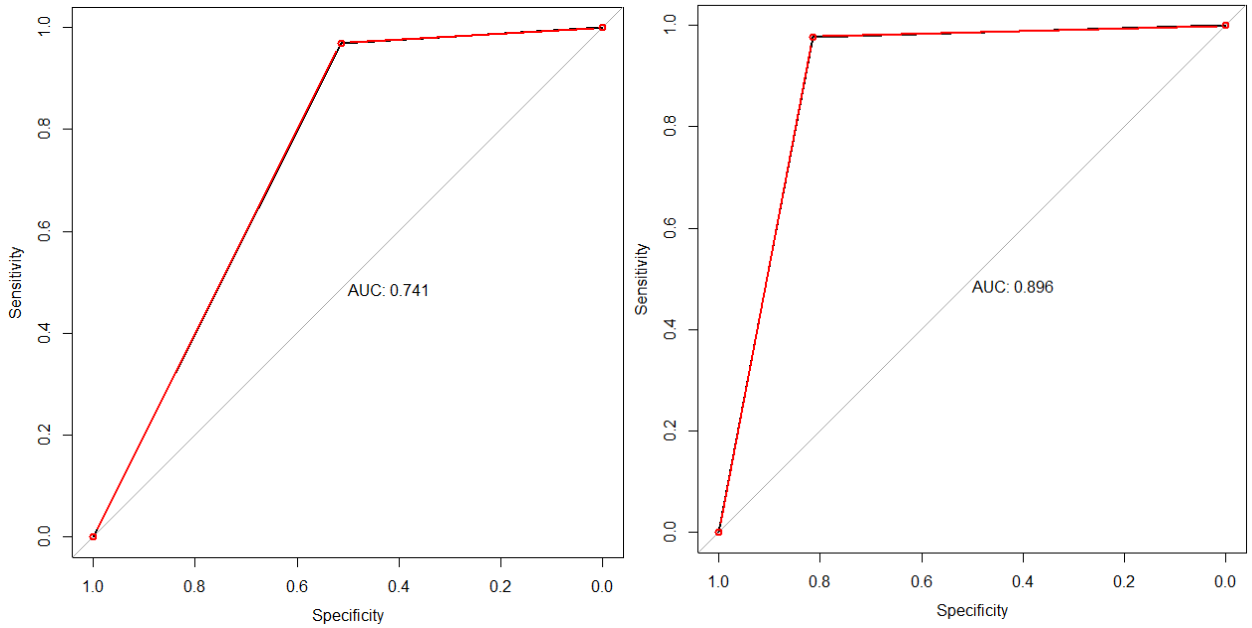


Рисунок 4.13 – ROC-криві моделей стекінгу та беггінгу

Отже, спочатку проаналізуємо результати роботи базових класифікаторів та роботи ансамблевих моделей першого та другого рівнів.

Таблиця 4.1 – Результати роботи базових класифікаторів

Тип моделі	Коефіцієнт успішності	Каппа-статистика	Чутливість (sensitivity)	Специфічність	Точність (precision)	Повнота (recall)	F міра (F-measure)	ROC-крива (AUC ROC)
Decision Tree	0,7	0,44	0,51	0,98	0,97	0,51	0,67	0,75
NB	0,67	0,36	0,59	0,8	0,81	0,59	0,68	0,69
LDA	0,64	0,27	0,68	0,59	0,71	0,68	0,69	0,63

Тип моделі	Коефіцієнт успішності	Каппа-статистика	Чутливість (sensitivity)	Специфічність	Точність (precision)	Повнота (recall)	Fміра (F-measure)	ROC-крива (AUC ROC)
QDA	0,67	0,38	0,45	0,98	0,97	0,45	0,61	0,72
LR	0,6	0,27	0,56	0,7	0,58	0,6	0,57	0,64
SVM	0,67	0,35	0,6	0,77	0,79	0,67	0,68	0,68
KNN	0,6	0,17	0,67	0,5	0,66	0,05	0,67	0,28
ANN	0,33	-0,45	0,05	0,44	0,04	0,63	0,04	0,58
RF	0,67	0,34	0,63	0,72	0,96	0,51	0,69	0,68

З таблиці 4.1 можна побачити, що найкращу оцінку специфічності та точності мають дерева рішень та метод квадратичного дискримінантного аналізу, найвищу точність мають дерева рішень. F-міра має найвищій показник для результатів для результатів випадкового лісу та лінійного дискримінантного аналізу, а площа під ROC-кривої найбільша для результатів класифікації з використанням дерев рішень. Загалом, базові класифікатори показали середній результат на даному наборі даних та результати потребують покращення. Найгірші результати дає штучна нейронна мережа.

Таблиця 4.2 – Результати роботи першого рівня моделі

Тип моделі	Коефіцієнт успішності	Каппа-статистика	Чутливість (sensitivity)	Специфічність	Точність (precision)	Повнота (recall)	Fміра (F-measure)	ROC-крива (AUC ROC)
Decision Tree	0,7	0,44	0,51	0,98	0,97	0,51	0,67	0,75
NB	0,67	0,36	0,59	0,8	0,81	0,59	0,68	0,69
QDA	0,67	0,38	0,45	0,98	0,97	0,45	0,61	0,72
LR	0,6	0,27	0,56	0,7	0,58	0,6	0,57	0,64
SVM	0,67	0,35	0,6	0,77	0,79	0,67	0,68	0,68

Тип моделі	Коефіцієнт успішності	Каппа-статистика	Чутливість (sensitivity)	Специфічність	Точність (precision)	Повнота (recall)	Fміра (F-measure)	ROC-крива (AUC ROC)
RF	0,67	0,34	0,63	0,72	0,96	0,51	0,69	0,68
Stacking LR	0,7	0,44	0,51	0,96	0,96	0,51	0,67	0,74

З таблиці 4.2 можна побачити, що використовуючи стекінг для комбінування шести не дуже високих результатів базових класифікаторів, трохи покращено загальний результат. Навчання моделі стекінгу відбувалося шляхом використання логістичної регресії *(glm)*.

Таблиця 4.3 – Результати роботи другого рівня моделі

Тип моделі	Коефіцієнт успішності	Каппа-статистика	Чутливість (sensitivity)	Специфічність	Точність (precision)	Повнота (recall)	Fміра (F-measure)	ROC-крива (AUC ROC)
Stacking LR	0,7	0,44	0,51	0,96	0,96	0,51	0,67	0,74
LDA	0,64	0,27	0,68	0,59	0,71	0,68	0,69	0,63
KNN	0,6	0,17	0,67	0,5	0,66	0,05	0,67	0,28
ANN	0,33	-0,45	0,05	0,44	0,04	0,63	0,04	0,58
Bagging	0,88	0,77	0,81	0,98	0,98	0,81	0,89	0,9

З таблиці 4.3 можна побачити, що використовуючи беггінг для комбінування трьох не дуже високих результатів базових класифікаторів та результату роботи моделі першого рівня – стекінгу, значно покращено загальний результат по всім метрикам оцінювання. Навчання моделі беггінгу відбувалося шляхом використання беггінгу дерев рішень *(treebag).*

Висновки до розділу 4

1. У якості базових моделей обрано моделі: дерева рішень (Decision Tree), Наївний Баєсів класифікатор (NB), лінійний дискримінантний аналіз (LDA),

квадратичний дискримінантний аналіз (QDA), логістична регресія (LR), метод опорних векторів (SVM), метод найближчого сусіда (KNN), штучні нейронні мережі (ANN) та модель випадкового лісу (RF).

2. Найкращу оцінку специфічності та точності мають дерева рішень та метод квадратичного дискримінантного аналізу, найвищу точність мають дерева рішень. F-міра має найвищій показник для результатів для результатів випадкового лісу та лінійного дискримінантного аналізу, а площа під ROC-кривої найбільша для результатів класифікації з використанням дерев рішень. Загалом, базові класифікатори показали середній результат на даному наборі даних та результати потребують покращення. Найгірші результати дає штучна нейронна мережа.

3. Використовуючи стекінг для комбінування шести не дуже високих результатів базових класифікаторів, трохи покращено загальний результат. Навчання моделі стекінгу відбувалося шляхом використання логістичної регресії *(glm)*.

4. Використовуючи беггінг для комбінування трьох не дуже високих результатів базових класифікаторів та результату роботи моделі першого рівня – стекінгу, значно покращено загальний результат по всім метрикам оцінювання. Навчання моделі беггінгу відбувалося шляхом використання беггінгу дерев рішень *(treebag)*.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було використано методи інтелектуального аналізу даних та інтелектуального прийняття рішень на основі даних навчальної вибірки. Також, розглянуто та проаналізовано одні з найбільш впроваджених з тих, що існують на даний момент, сучасних методів прогнозування, що базуються на деревах рішень, наївному баєсівському класифікаторі, лінійному дискримінантному аналізі, квадратичному дискримінантному аналізі, логістичній регресії, методі опорних векторів, методі найближчого сусіда, штучних нейронних мережах та моделі випадкового лісу.

Для покращення результатів прогнозування використано ансамблеві методи – потужний інструмент для побудови моделей машинного навчання. Кілька базових моделей навчались для вирішення однієї і тієї ж проблеми та було об'єднано для отримання кращих результатів. Проведено дослідження базових методів прогнозування та ансамблевих моделей на основі стекінгу та беггінгу, а також метрик оцінки ефективності використання базових класифікаторів та моделей першого та другого рівня, визначено наступні параметри для усіх наведених методів у роботі: точність прогнозування та коефіцієнт помилок, Каппа-статистика, чутливість та специфічність, точність та повнота, F-міра та площею під ROC-кривою.

Метою магістерської кваліфікаційної роботи була розробка та реалізація інтелектуальної системи класифікації сервісів E-Commerce компанії на основі ансамблів моделей для вивчення споживчих запитів. Виявлено чинники, пов'язані з ризиком невчасної доставки товару клієнту.

Виходячи з мети, виконано поставлені завдання:

1) вивчено та досліджено методи ансамблювання моделей, засоби підготовки даних та метрик якості,

2) обрано алгоритм та побудовано базові моделі,

3) розроблено перший шар, що складається з набору базових моделей,

4) оброблено результати моделей першого шару як входи моделі другого шару(ансамбль),

5) проаналізовано результати та створено ансамбль моделей,

6) проаналізовано ефективність ансамблевих моделей.

Метрики оцінки якості роботи базових класифікаторів показали опосередковані результати, що потребують покращення**.** Використовуючи стекінг для комбінування шести не дуже високих результатів базових класифікаторів, трохи покращено загальний результат. Використовуючи беггінг для комбінування трьох не дуже високих результатів базових класифікаторів та результату роботи моделі першого рівня – стекінгу, значно покращено загальний результат по всім метрикам оцінювання.

В першому розділі здійснено огляд та класифікацію інтелектуальних систем та їх можливостей для вирішення задач прогнозування, а також, використання задач класифікації у керуванні компанією.

У другому розділі описано ключові характеристики якості моделей, вибір метрики, підбір базових (слабких) моделей, підбір параметрів для базових моделей та ансамблевих методів.

В третьому розділі описано процес збору даних, їх аналіз та інтерпретацію, виконано попередня обробка даних та розділено набір на тренувальну та тестову вибірки й згенеровано вхідні змінні на основі поведінкових даних клієнтів.

У четвертому розділі наведено результати застосування простих класифікаторів та ансамблю моделі дворівневої класифікації та виконано оцінку ефективності розроблених моделей класифікації.

У спеціальній методичній частині магістерської кваліфікаційної роботи було використано навички з дисципліни «Методи та системи машинного навчання» для створення практичних робіт. Було створено практичну роботу на тему *«Створення моделей нейронних мереж для вирішення задач прогнозування»*. Визначено мету, завдання практичної роботи. Наведено короткі теоричні відомості з досліджуваного питання та приклад виконання практичного завдання.

У спеціальній частині з охорони праці та безпеки при надзвичайних ситуаціях було описано робоче місце та розглянуто умови праці у ньому. Виконано розрахунки щодо оснащення приміщення шумоізоляціонними панелями для запобігання зависокого рівня шуму на робочому місці. Розглянуто забезпечення безпеки персоналу в умовах надзвичайних ситуацій, пов’язаних з повітряною тривогою та ядерним ударом.

Використовуючи ансамбль всіх згаданих моделей нам вдалося використати сильні сторони кожної з них і отримати бажаний результат.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зайченко Ю. П. Основи проектування інтелектуальних систем. Київ: Слово, 2006. 352 с.

2. Береза А.М. Основи створення інформаційних систем: навч. посібник. Київ: КНЕУ, 2001. 205 с.

3. Любарський С.В., Шаціло П.В. Методологія вибору моделі подання знань в інтелектуальних навчальних системах: збірник наукових праць ВІТІ НТУУ «КПІ». Київ, 2010. С. 65–71.

4. Боюн В. П. Інтелектуальні комп'ютерні системи сприйняття і обробки фізичної інформації: вісник Національної академії наук України. Київ, 2015. С. 82–84.

5. Вергунов В. В. Інтелектуальні системи збереження даних як інструмент для підтримки оптимальної структури даних та архітектури бази даних: вісн. КНУ ім. Тараса Шевченка. Київ, 2013. С. 122–127.

6. Данчук В. Д., Лемешко Ю. С., Лемешко Т. А. Інтелектуальні інформаційні системи управління науковими проектами. Київ: НТУ, 2013. С. 21–27.

7. Фадєєва І. Г. Інтелектуальні технології у фінансовому інжинірингу: міжнар. наук.-вироб. журн. Сталий розвиток економіки. Хмельницький, 2014. С. 234–242.

8. Гужва В. М., Скрипова О. С. Інтелектуальні системи підтримки прийняття рішень у страхуванні: потреби українських страхових компаній та їх задоволення. Київ: КНУБА, 2012. С. 183–187.

9. Лізунов П. П., Івлева Н. П., Васильєва Г. Л. Інформаційні системи в менеджменті: навчальний посібник. Київ: КНУБА, 2010. 128 с.

10. Філімонова О. Ю., Васильєва Г. Л. Інформаційні системи в менеджменті: методичні вказівки. Київ: КНУБА, 2011. 44 с.

11. Ульянченко О.В. Дослідження операцій в економіці: підручник. Суми: Довкілля, 2010. 594 с.

12. Ілляшенко, С.М. Інноваційний розвиток: маркетинг і менеджмент знань: монографія. Суми: ДісаПлюс, 2016. 192 с.

13. Сальніков О.М., Романюк В.А., Оленченко В.Т. Інформаційні системи в менеджменті. Теоретичні основи інформаційних систем в менеджменті: навчальний посібник. Харків: Національна академія Національної гвардії України, 2015. 203 с.

14. Юдін О. М., Макарова М. В., Лавренюк Р. М. Системи електронної комерції: створення, просунення і розвиток: монографія. Полтава : РВВ ПУЕТ, 2011. 201 с.

15. Ставерська Т. О., Андрющенко І. С. Фінансове планування та прогнозування в підприємствах і фінансових установах: навч. посібник ХДУХ. Харків, 2013. 146 с.

16. Біла О. Г. Фінансове планування і прогнозування: навч. посіб. Львів: Компакт–ЛВ, 2005. 312 с.

17. Кузнєцова Н.В. Методи і моделі аналізу, оцінювання та прогнозування ризиків у фінансових системах: дис. д.т.н.: 01.05.04. Київ, 2018. 415 с.

18. Understanding the Bias–Variance Tradeoff URL: http://scott.fortmannroe.com/docs/BiasVariance.html (дата звернення: 20.12.2022).

19. Гожий О.П., Калініна І.О. Методичні рекомендації до виконання курсової роботи з дисципліни «Методи та системи машинного навчання». Миколаїв: ЧНУ ім. П. Могили, 2021. 17 с.

20. Stephen Marsland. Machine Learning: An Algorithmic Perspective. Palmerston North: Massey University, 2015. 452 p.

21. Liang Wang, Li Cheng, Guoying Zhao. Machine Learning for Human Motion Analysis. Anhui: IGI Global, 2009. 318 p.

22. Литвин А.В. Розробка дерев рішень для прогнозування фінансової кризи в страхових компаніях України. Київ: Наукові записки НаУКМА, 2015. С. 59–64.

23. Субботін С.О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень. Запоріжжя: ЗНТУ, 2008. 341 с.

24. Schapire R.E. The strength of weak learnability. Machine Learning, Vol. 5, No. 2. Boston, 1990. P. 197–227.

25. Hastie, T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. California: Springer–Verlag, 2009. 746 p.

26. Дубініна С. В. Байєсівські методи моделювання актуальних процесів та оцінювання ризиків страхових компаній: дис. канд. техн. наук: 05.13.23. Київ, 2017. 199 с.

27. Ethem Alpaydin. Introduction To Machine Learning. 3th ed. Massachusetts, 2009. 584 p.

28. Глибовець М. М., Олецький О.В. Штучний інтелект: підручник для студ. вищих навч. закладів. Київ: КМ Академія, 2002. 369 с.

29. Загоруйко Н. Г. Прикладные методы анализа данных и знаний. Новосибирск: ИМСО РАН, 1999. 270 с.

30. Raviv Y., Intrator N. Bootstrapping with noise: An effective regularization technique. Connection Science 8 (3-4). Tel Aviv, 1996. P. 355-372.

31. Нікольский Ю.В., Пасічник В.В., Щербина Ю.М. Системи машинного навчання. Львів, 2010. 282 с.

32. Чернишева Д.С., Будник М.М. Застосування дискримінантного аналізу до обробки магнітокардіографічної інформації: збір. наук. праць "Комп'ютерні засоби, мережі та системи". Київ: Інститут кібернетики імені В. М. Глушкова НАН України, 2004. С. 57–64.

33. Бідюк П.І., Кузнєцова Н.В., Терентьєв О.М. Система підтримки прийняття рішень для аналізу даних. Київ: Наукові вісті НТУУ «КПІ», 2011. С. 48–61.

34. Дудка Б.Р., Бідюк П.І. Реалізація методики побудови моделей часових рядів. Системний аналіз та кібернетика. Київ, 2016. С.233–255.

35. Невмержицький О.В. Аналіз сучасних моделей, орієнтованих на знання, та методів прийняття рішень. Київ: КНУБА, 2013. С. 119–125.

36. Christopher M Bishop. Pattern recognition and Machine Learning. Cambridge, 2006. 128 p.

37. Використання логістичної регресії для визначення критеріїв постановки діагнозу реактивного або хронічного панкреатиту / Сіткар А. Д., Ростока Л. М., Немеш І. М., Лигирда, О. В. Ужгород, 2019. С. 235–236.

38. Шеремет О.І., Садовой О. В. Метод опорних векторів (SVM): наук. журнал “Математичні моделі“. Донецьк, 2013. С. 13–17.

39. Zhan Zh. Introduction to machine learning: k-nearest neighbors. Vienna: Ann Transl Med, 2016. 218 p.

40. Кононюк А.Ю. Нейроні мережі і генетичні алгоритми. Київ: «Корнійчук», 2008. 446 с.

41. Калініна І. О. Дослідження нейромережевих методів у задачах прогнозування: Науково–методичний журнал. Миколаїв: ЧНУ ім. П. Могили, 2009, С. 132-138.

42. Hansen L., Salamon P. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence. San Diego, 1990. P. 993–1001.

43. Breiman, L. Random forests. Machine Learning: technical report 518. California: UC Berkeley, 2001. P. 5–32.

44. Maniruzzaman M., Rahman MJ., Ahammed B. Classification and prediction of diabetes disease using machine learning paradigm. Health information science and systems, Vol. 8. Texas, 2020. P. 1–14.

45. Artificial Intelligence: A Modern Approach. URL: https://towardsdatascience.com/understanding–the–bias–variance–tradeoff (дата звернення: 20.12.2022).

46. Кузьменко Б. В., Чайковська О. А. Системи штучного інтелекту: навч. посібник. Київ: Альтерпрес, 2006. 140 с.

47. Снитюк В. Є. Прогнозування. Моделі. Методи. Алгоритми: навч. посіб. Київ: Маклаут, 2008. 364 с.

48. Opitz D., Maclin R. Popular ensemble methods: An empirical study: journal of Artificial Intelligence Research No. 11. El Segundo, 1999. P. 169–198.

49. Ensemble Methods to Optimize Machine Learning Models. URL: https://hub.packtpub.com/ensemble–methods–optimize–machine–learning–models (дата звернення: 20.12.2022).

50. Бармак О. В., Крак Ю. В., Манзюк Е. А. Характеристика для вибору моделей у ансамблі класифікаторів: науковий журнал “Проблеми програмування”. Київ, 2018. С. 171–179.

51. Великоіваненко Г.І., Савіна С.С. Побудова ансамблів моделей кредитного скорингу. Київ: КНЕУ ім. Вадима Гетьмана, 2018. 44 с.

52. Dietterich T., Ensemble Methods in Machine Learning. URL: http://web.engr.oregonstate.edu/~tgd/publications/mcs–ensembles.pdf (дата звернення: 25.12.2022).

53. Zhi–Hua Zhou Ensemble Learning. URL: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/springerEBR09.pdf (дата звернення: 20.12.2022).

54. Бустінг і беггінг як методи формування ансамблей моделей / Вербівський Д. С., Карплюк, С. О., Фонарюк, О. В., Сікора, Я. Б. Житомир: ЖДУ ім. Івана Франка, 2021. С. 163-169.

55. Метрики в задачах машинного навчання. URL: https://habr.com/ru/company/ods/blog/328372 (дата звернення: 20.12.2022).

56. Ставицький А.В. Класифікаційні метрики. URL: http://www.andriystav.cc.ua/Downloads/MITER/Lecture_04.pdf (дата звернення: 20.12.2022).

57. Improvements on Cross–Validation: The 632+ Bootstrap Method. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1997.10474007#.U2o7MVdMzTo (дата звернення: 22.12.2022).

58. Ensemble methods: bagging, boosting and stacking. URL: https://towardsdatascience.com/ensemble–methods–bagging–boosting–andstacking–c9214a10a205 (дата звернення: 27.12.2022).

59. Breiman L. Bagging predictors. Machine Learning: technical report 421. California: UC Berkeley, 1996. P. 123–140.

60. Freund Y., Schapire R. A decision–theoretic generalization of online learning and an application to Boosting. Journal of Computer and System Sciences 55. Barcelona, 1997. P. 119–139.

61. Wolpert D.H. Stacked generalization. Neural Networks, Vol. 5, No. 2. Boston, 1992. P. 241-259.

62. Ting K., Witten I. Issues in stacked generalization: journal of Artificial Intelligence Research No.10. El Segundo, 1999. P. 271–289.

63. E–Commerce Shipping DataSet. URL: https://www.kaggle.com/datasets/+ prachi13/customer-analytics (дата звернення: 15.11.2022)

64. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 10.01.2023)

65. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин. URL: https://zakon.rada.gov.ua/laws/show/z0382-99#Text (дата звернення: 10.01.2023)

66. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 11.01.2023)