math_2023_057

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики
кафедра математичного моделювання

ДОСЛІДЖЕННЯ МОДЕЛЕЙ КЛАСИФІКАЦІЇ У СЕРЕДОВИЩІ R

Кваліфікаційна робота

Рівень вищої освіти – перший (бакалаврський)

Виконав:
студент 4 курсу, 407 групи
Нестор Миронюк
Керівник:
кандидат фізико-математичних наук,
доцент Дорошенко І. В.

До захисту допущено
на засіданні кафедри
протокол №__ від ___ _____ 2023 р.
Зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2023

Анотація

У даній кваліфікаційній роботі проведено порівняння алгоритмів класифікації даних, які є представниками алгоритмів машинного навчання з учителем, на основі акуратності отриманих моделей. Побудову моделей класифікації здійснено з використанням реальних та симульованих методом Монте-Карло даних. Алгоритми машинного навчання, представлені в роботі: Support Vector Machine, Random Forests, K Nearest Neighbour, Linear Discriminant Analysis, Quadratic Discriminant Analysis. Усі статистичні дослідження проведено у середовищі R Programming з надбудовою RStudio.

Ключові слова: класифікація даних, Support Vector Machine, Random Forests, K Nearest Neighbour, Linear Discriminant Analysis, Quadratic Discriminant Analysis.

Annotation

In this qualification work, a comparison of data classification algorithms, which are representatives of machine learning algorithms with a teacher, was made, based on the accuracy of the obtained models. Construction of classification models was carried out using real and simulated Monte Carlo data. Machine learning algorithms presented in the work: Support Vector Machine, Random Forests, K Nearest Neighbor, Linear Discriminant Analysis, Quadratic Discriminant Analysis. All statistical studies were performed in the R Programming environment with the RStudio add-on.

Ключові слова: data classification, Support Vector Machine, Random Forests, K Nearest Neighbour, Linear Discriminant Analysis, Quadratic Discriminant Analysis.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ Миронюк Нестор
(підпис)

ЗМІСТ

Вступ

Класифікація даних у широкому сенсі визначається як процес упорядкування даних за відповідними категоріями, щоб вони могли використовуватися та захищатися більш ефективно. На базовому рівні процес класифікації полегшує пошук і отримання даних. Класифікація даних має особливе значення, коли йдеться про управління ризиками, відповідність та безпеку даних. Класифікація даних передбачає позначення даних тегами для полегшення їх пошуку та відстеження. Це також усуває багаторазове дублювання даних, що може зменшити витрати на зберігання та резервне копіювання, одночасно прискорюючи процес пошуку.

За останні два десятиріччя класифікація даних значно покращилася. Сьогодні ця технологія використовується для різноманітних цілей, часто для підтримки ініціатив із захисту даних. Але дані можуть бути засекреченими з кількох причин, включаючи легкість доступу (при уникненні несанкціонованого доступу), підтримання нормативної відповідності та для задоволення різноманітних інших ділових чи особистих цілей. У деяких випадках класифікація даних є нормативною вимогою, оскільки дані мають бути придатними для пошуку та відновлення протягом визначених часових проміжків. З метою безпеки даних класифікація даних є корисною тактикою, яка сприяє належним реакціям безпеки на основі типу даних, які витягуються, передаються або копіюються. Класифікація даних часто включає в себе безліч тегів і міток, які визначають тип даних, їх конфіденційність і цілісність.

Отже, класифікація даних у наш час є широко застосовною у багатьох сферах життєдіяльності людини, наприклад:

- у маркетингу – класифікація товарів та послуг для різного роду користувачів;

- у економіці – класифікація цінних паперів для вибору оптимального портфелю інвестором;
- у медицині – для встановлення правильного діагнозу та вибору найкращого методу лікування;
- у кібербезпеці – для запобігання шахрайських операцій.

За кожним із вказаних застосувань стоїть математичний алгоритм, за допомогою якого і відбувається певна класифікація даних. Звичайно, спеціаліст з певної галузі може самостійно визначити клас, до якого потрібно віднести певну подію з відповідними характеристиками. Уявімо тепер, що таких даних тисячі чи, навіть, мільйони. Тут на допомогу і приходять машинне навчання, яке здатне робити висновки, опираючись на доступні набори даних. У даній роботі покроково описано роботу кожного з п’яти алгоритмів класифікації даних та проілюстровано результати їхньої роботи на реальних наборах даних.

Метою роботи є дослідження, практичне застосування та порівняння існуючих алгоритмів класифікації даних.

Об’єктом дослідження є алгоритми класифікації даних.

Предмет дослідження – застосування існуючих алгоритмів класифікації реальних даних.

Основні завдання роботи:

- ознайомитися з існуючими алгоритмами класифікації даних, які є представниками машинного навчання з учителем;
- порівняти кілька сучасних методів класифікації даних на прикладі реальних даних;
- навчитися будувати класифікатори у середовищі R Programming з використанням пакету-надбудови RStudio; аналізувати результати роботи відповідних класифікаторів.

РОЗДІЛ І. ЗАДАЧІ КЛАСИФІКАЦІЇ ДАНИХ

1.1. Історія розвитку класифікації даних

Michie, Spiegelhalter і Taylor (1994) як редактори представили у своїй книзі «Машинне навчання, нейронна та статистична класифікація» основні історичні підходи дослідження в галузі класифікації (статистичні – машинне навчання та нейронні мережі). Вони описали, як дослідники намагалися вивести певний вид класифікаторів, які були б здатні відповідати поведінці людини, яка приймає рішення. До того ж, ці класифікаори водночас мали бути послідовними, вирішувати широкий спектр проблем класифікації та використовувалися в реальних життєвих ситуаціях.

Warren Sarle (1994) запропонував штучні нейронні мережі з іншої точки зору, а саме – статистичний підхід до впровадження нейронних мереж. Він також описав, як штучні нейронні мережі можуть обробляти величезні обсяги даних настільки точно, як багато інших статистичних методів. У своїй роботі Sarle показав зв’язок між нейронними мережами та статистичними моделями, такими як узагальнені лінійні моделі, поліноміальна регресія, непараметрична регресія та кластерний аналіз.

Steven Salzberg (1997) представив порівняльне дослідження деяких методів класифікації. Він зосередився на якості планів експериментів і тому, як це може призвести до статистично недійсних висновків, якщо необхідні експерименти не були розроблені дуже ретельно. У своїй роботі Salzberg описав кілька випадків, які можуть зробити експериментальне порівняння недійсним, якщо їх проігнорувати. Крім того, він також довів, що ці випадки та їхні висновки застосовуються до класифікації, а також до обчислювальних експериментів у класифікаційних полях.

Mohd Yusoff Mashor (2000) представив важливе порівняння між зворотним поширенням, рекурсивною помилкою передбачення та модифікованими алгоритмами рекурсивної помилки передбачення для навчання багаторівневих

мереж перцептронів. У своєму дослідженні він досліджував продуктивність цих алгоритмів і проводив експерименти в реальному житті, використовуючи реальні дані, щоб підтвердити свої висновки.

Kenji Fukumizu (2000) вирішив проблему активного навчання в багатошарових персептронах, оскільки продуктивність нейронних мереж значно покращується, якщо навчальні дані вибираються активно. Він представив спосіб підготовки розподілу ймовірностей за допомогою запропонованого методу активного навчання. У результаті він отримав зразки навчальних даних із зазначеного розподілу. Основою цієї методології була розробка критерію на основі інформаційної матриці.

Jiawei Han і Micheline Camber (2001) представили у своїй книзі «Інтелектуальний аналіз даних: концепції та методи» деякі методи класифікації, які підходять для технології інтелектуального аналізу даних. Вони об'єднали процес класифікації та процес прогнозування, щоб розглядати їх як єдиний процес у багатьох сучасних компаніях. Вони визначили, що процес класифікації даних насправді складається з двох пов'язаних фаз, а саме фази навчання та фази тестування.

Josef Reisinger, Kenneth O. StanleyRisto Miikulainen (2004) працювали над багатообіцяючим підходом до нейронних мереж, який автоматично змінює топологію та ваги мережі. Показано, що цей підхід є потужним у проблемах класифікації нелінійної оптимізації [17].

1.2. Поняття алгоритму класифікації даних

Класифікація даних використовується як у математичних задачах, так і в реальному житті. Загалом, цей термін можна використовувати для будь-якої діяльності, яка виводить певне рішення чи прогноз на основі наявної на даний момент інформації. Використовуючи більш точне визначення, процедура класифікації – це побудова певного типу методу для винесення суджень щодо

неперервної послідовності випадків, де кожен новий випадок повинен бути віднесений до одного з попередньо визначених класів. Цей тип побудови було названо навчанням під наглядом (Suppervised Learning), щоб відрізнити його від навчання без нагляду (Unsupervised Learning) або кластеризації, у якому класи не визначені заздалегідь, а виводяться на основі доступних даних.

Інформація є основою сучасного бізнесу. Різні види даних, такі як клієнтські бази даних, бази даних інвентаризації, медичні та статистичні бази даних, підтримують мінливі вимоги сучасної економіки. Оскільки обсяг даних швидко зростає з кожним днем, організації повинні мати потужні та ефективні інструменти опрацювання даних. У наш час розуміння цінності наших даних вважається ключовим моментом розвитку бізнесу та наукових досліджень.

Бази даних повні прихованої та багатої інформації, яку можна використовувати для покращення бізнес-рішень. Класифікація – це тип аналізу даних, який можна використовувати для виділення функцій, що описують важливі класи даних. Класифікація даних – це процес пошуку набору моделей, які розрізняють і диференціюють класи даних, з метою прогнозування класу заданих зразків даних, мітка класу яких невідома. Отримана модель повністю базується на аналізі навчального набору даних, який містить зразки даних, мітка класу яких уже відома. Похідна модель може бути представлена в багатьох формах, таких як нейронні мережі, математичні формули або дерева рішень.

Класифікація даних – це процес, який складається з двох етапів [2]:

- На першому кроці виводиться модель, яка описує заздалегідь визначений набір класів даних; модель будується шляхом аналізу записів навчального набору даних, які, у свою чергу, описуються окремими атрибутами або функціями. Передбачається, що кожен запис належить до попередньо визначеного класу; він визначається атрибутом мітки класу. Зразки в навчальному наборі даних вибираються випадковим чином із простору об’єктів. Оскільки мітка

класу кожного навчального зразка вже відома та надається в наборі даних, цей крок також відомий як контрольоване навчання. У випадку, якщо мітка класу не надається, крок може бути відомий як неконтрольоване навчання або кластеризація. Після побудови моделі на першому кроці виконується другий крок.

- На другому кроці модель використовується для початку фактичного процесу класифікації. Цей процес застосовується до тестового набору даних, який містить решту зразків із простору об’єктів. Для кожного тестового зразка дана мітка класу, яка представляє бажаний результат, порівнюється з вивченим результатом або фактичним результатом для цього зразка.

Різні методи класифікації були запроваджені багатьма дослідниками в різних галузях, таких як машинне навчання, експертні системи, статистика та медицина. Більшість алгоритмів класифікації та навчання є резидентними, оскільки припускають невеликий розмір даних. Сьогодні широко запропоновані більш просунуті методи, які вважаються масштабованими для обробки великих резидентних даних. Оскільки останні проблеми класифікації стають все більш складними та містять дуже великі дані, ці нові методи відіграють дуже важливу роль у створенні масштабованих та ефективних систем класифікації.

Важливим компонентом багатьох методів аналізу даних є пошук хорошого та ефективного алгоритму класифікації; цей процес вимагає дуже ретельного планування, щоб максимізувати шанси на успіх. Різні критерії глибоко заважають знайти алгоритм класифікації, який допомагає вирішити дану задачу класифікації. Розмір сховищ даних, складність заданих проблем і період часу, необхідний для завершення процесу навчання, є одними з цих важливих критеріїв. Кожен із цих алгоритмів класифікації може підходити для ряду певних проблем; але не всі ці алгоритми можна використовувати для вирішення будь-якої проблеми класифікації.

Класифікація даних особливо корисна, коли певне сховище даних містить приховану інформацію, яку можна використовувати для прийняття майбутніх рішень; наприклад, для медичної діагностики, для аналізу наукових даних або для статистичного аналізу. Дослідники мають у своєму розпорядженні широкий спектр різноманітних алгоритмів класифікації, включаючи техніку найближчого сусіда (nearest neighbor technique), індукцію дерева рішень (decision tree induction) (мал. 1), алгоритм навчання персептронів (perceptron learning algorithm) (мал. 2) і зворотне поширення помилок (error backpropagation). За останні роки було введено багато оновлень цих алгоритмів. Ці оновлення були спрямовані на підвищення ефективності цих алгоритмів. Таким чином, сьогоднішні дослідження стикаються з основною проблемою використання цих алгоритмів, а саме як можна вибрати найкращий алгоритм, який буде реалізовано для ефективного вирішення нової проблеми класифікації.

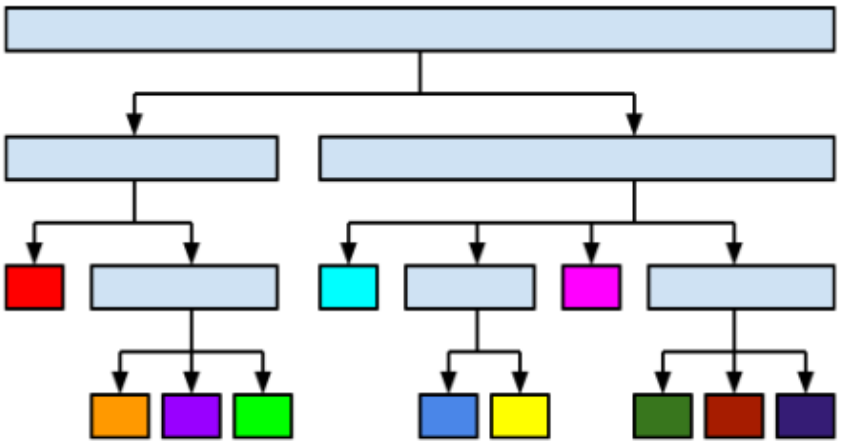


Рисунок 1. Класифікація даних у вигляді дерева рішень

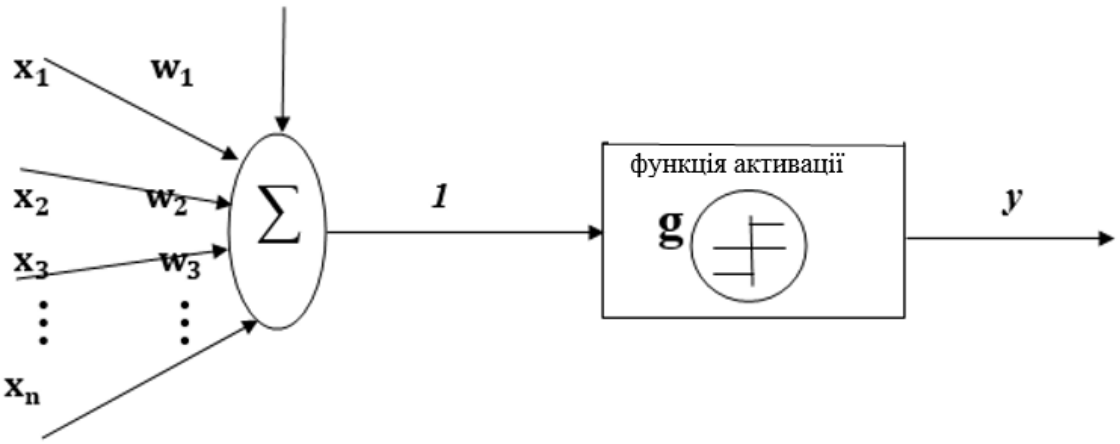


Рисунок 2. Архітектура одношарового персептрона

Індекс класифікації даних формується відповідно до реальної ситуації та потреб підприємства. Він класифікується за системою класифікації відповідно до стандартів даних і, нарешті, поділяє користувачів різних типів, різних вікових груп і різних цілей на кілька підкатегорій. Основною метою класифікації даних є ефективна класифікація різних показників діяльності підприємства.

Алгоритм класифікації даних на основі частоти доступу визначає частоту доступу до даних шляхом запису кількості звернень до даних за певний період часу, а потім класифікує дані відповідно до частоти доступу. В даний час алгоритми класифікації даних на основі частоти доступу в основному включають метод фіксованого порогу.

Цей метод встановлює фіксоване закрите значення частоти доступу до даних як стандарт класифікації даних, щоб реалізувати класифікацію даних. Основний процес полягає в наступному [14]:

1) Запустити умову міграції та підрахувати кількість відвідувань даних у розширеному сховищі протягом періоду часу.
2) Обчислити частоту доступу до даних G за період часу T:

$$\frac{"a}{G} = T$$

3) Встановити поріг доступу до фіксованої частоти $Gmax$.

4) Для даних G $< Gmin$, до яких часто звертаються в межах T, вони додаються до черги очікування міграції та переносяться до сховища нижчого рівня.
5) Повторити процес.
6) Алгоритм завершується.

У аналізі даних ми визначаємо класифікацію як: класифікація полягає в отриманні цільової функції шляхом навчання. Дослідження класифікації потоків даних мають два основні напрямки, а саме алгоритми інкрементального навчання та алгоритми ансамблевого навчання.

Для покращення ефективності базової комбінації класифікаторів використовуються методи повторних зразків. Рекламна акція додає вагу кожному базовому класифікатору, і, нарешті, невідомі дані зважуються та оцінюються відповідно до відповідної ваги кожного класифікатора, щоб отримати результат остаточного рішення.

1.3. Технології аналізу даних

Система інтелектуального аналізу даних існує не одна, вона включає багато компонентів, таких як бази даних, файлові системи, алгоритми інтелектуального аналізу даних, системи аналізу та вихідні результати тощо. Алгоритми інтелектуального аналізу даних зазвичай використовуються для обробки даних, і основною функцією системи аналізу та виведення результатів є аналіз результатів обробки даних.

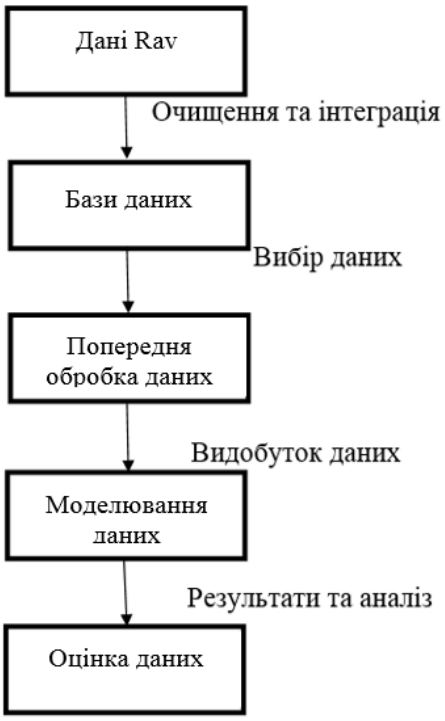


Рисунок 3. Процес інтелектуального аналізу даних

Класифікацію інтелектуального аналізу даних зазвичай поділяють на два етапи [29]:

(1) Фаза навчання даних: нам потрібно витягти частину даних як набір для навчання, а потім використати алгоритм класифікації даних, щоб створити класифікатор для набору даних.

(2) Етап класифікації даних: цей крок передбачає використання розробленого класифікатора для класифікації великої кількості даних.

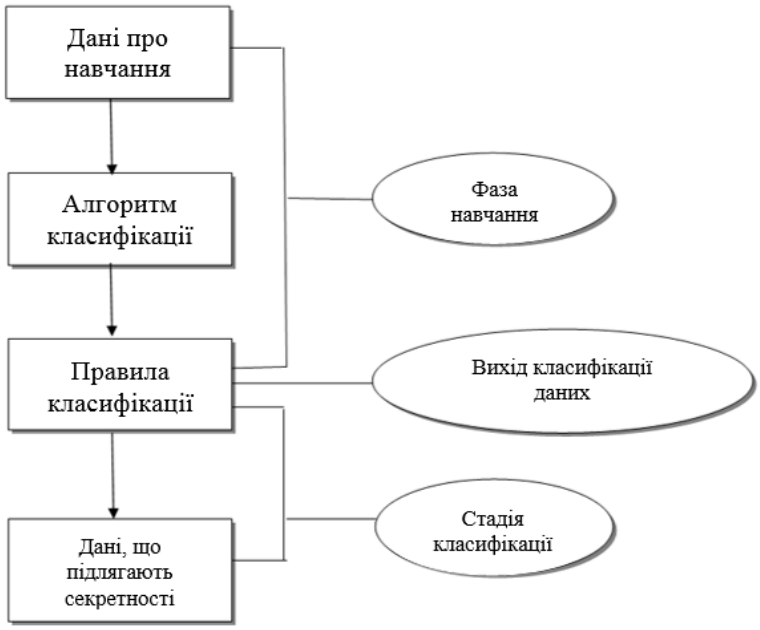


Рисунок 4. Процес класифікації інтелектуального аналізу даних

1.3.1. Теорії, пов'язані з незбалансованою класифікацією даних

Є багато факторів, які впливають на класифікацію незбалансованих даних, наприклад невідповідні стандарти оцінки та відсутні дані. Метод обробки на рівні даних полягає в повторній вибірці даних, і найбільш типовим алгоритмом є алгоритм SMOTE. Алгоритм SMOTE – це новий тип методу передискретизації, який може не тільки краще уникнути проблеми переобладнання, спричиненої класифікатором під час класифікації, але також дозволяє класифікатору мати більший простір для узагальнення для вибірок меншості. Більш поширені алгоритми на рівні алгоритму включають економічне навчання та алгоритм K-найближчого сусіда (K-nearest neighbor algorithm) [9].

Щоб краще реалізувати класифікацію та керування даними, система розділена на 3 рівні. Перший рівень – це серверна платформа бізнес-логіки, найнижче прикладне програмне забезпечення у великій корпоративній базі даних. Другий рівень полягає у створенні сервісної платформи аналізу даних на основі архітектури B/S у компанії. Третій рівень забезпечує пов’язані технології,

такі як розробка, проектування та підтримка функціональних модулів підтримки для клієнта, тобто клієнта як основного органу.

У розподіленій обчислювальній платформі моделювання необхідно зберігати два типи інформації:

- інформацію користувача
- інформацію моделювання.

Взаємодія між розподіленою обчислювальною платформою спільного моделювання та базою даних реалізується за допомогою інтерфейсу ADO. Викликаючи інтерфейс, наданий ADO, ви можете отримати доступ до даних, щоб додавати, видаляти, перевіряти та змінювати дані, збережені в базі даних. Цей інтерфейс зазвичай повертає набір записів або нульовий покажчик.

1.3.2. Технологія міграції даних на основі ієрархічного зберігання

- Онлайн-сховище: Інтернет-сховище має високу швидкість доступу та високу ціну. Онлайн-сховище зазвичай використовує технологію високопродуктивної системи зберігання високого класу з високою доступністю та резервуванням.
- Офлайн-сховище. Офлайн-сховище також відоме як резервне сховище. Його швидкість доступу низька, але він може досягти масового зберігання та нижчої вартості.
- Зберігання майже на одній лінії: зберігання рідко використовуваних або недоступних даних в низькопродуктивних пристроях зберігання [18].

РОЗДІЛ ІІ. МАШИННЕ НАВЧАННЯ З УЧИТЕЛЕМ

2.1. Етапи класифікації даних

Основна концепція класифікації даних полягає в тому, щоб визначити, до якого класу належить певна точка даних на основі характеристик, якими володіє ця точка. Це можливо шляхом порівняння з характеристиками, відомими для кожного з можливих класів, а потім дані класифікуються як клас, характеристики якого найбільше відповідають. Вимога, звичайно, полягає в тому, щоб інформація про різні класи була зібрана заздалегідь. Це робиться шляхом «навчання» моделі класифікатора з використанням набору даних, де точки даних у ньому були класифіковані вручну за різними класами. На Рис. 5 показано процес навчання класифікатора [22].

Рисунок 5. Модель побудувати моделі класифікації

Щоб навчити класифікатор, спочатку кожна з попередньо класифікованих точок даних із набору даних проходить через екстрактор ознак, який аналізує дані та зберігає ряд атрибутів, які можуть ідентифікувати цю конкретну точку даних. Оскільки відомо, яку класифікацію має ця точка даних, зібрані ознаки вставляються в алгоритм машинного навчання, який потім намагається зробити висновки на основі всіх зібраних класифікованих ознак, а потім створює на їх основі модель, яку можна використовувати для класифікаії немаркованих даних. Наочний приклад цього можна побачити на малюнку 5.

2.2. Тестування даних за допомогою класифікатора

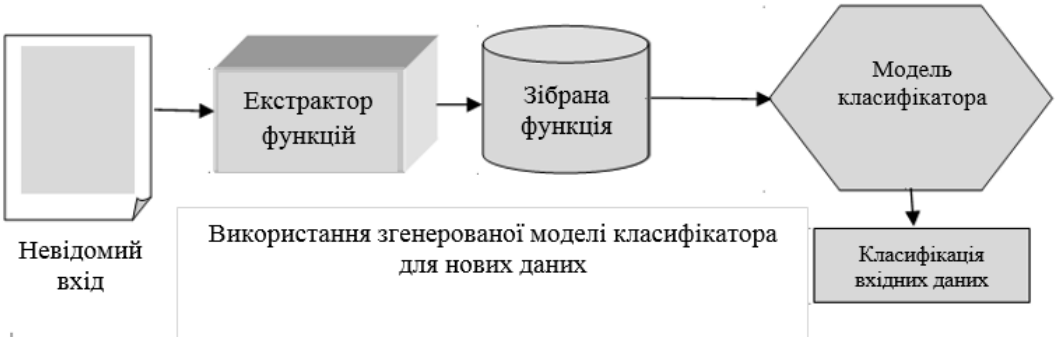


Рисунок 6. Невідомі дані класифікуються за допомогою попередньо створеної моделі класифікатора

Модель класифікатора, яка була згенерована з використанням попередньо позначених даних, може бути використана для класифікації невідомих вхідних даних за тим самим принципом. Кожна точка даних у наборі даних проходила через екстрактор ознак, який потім надсилався до моделі класифікатора. Потім ці немарковані об’єкти порівнювали з об’єктами в моделі класифікатора, а потім класифікували як клас, до якого вони були найбільш схожі. Наочний приклад цього можна побачити на Рис. 6.

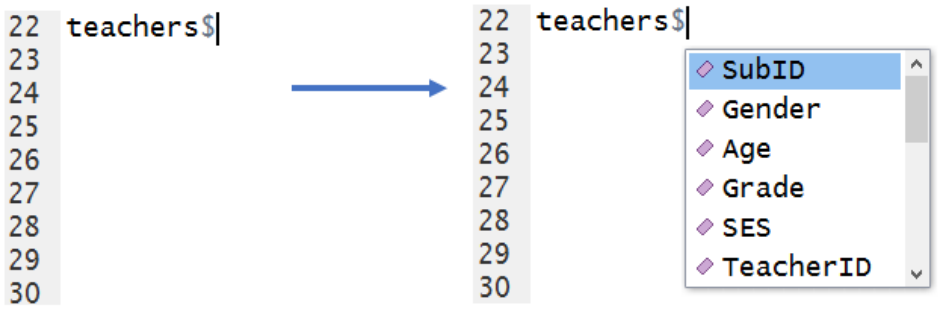
Теоретично використання великого набору даних для побудови моделі класифікатора підвищить продуктивність під час класифікації нових даних, оскільки було б легше побудувати більш загальну модель і, отже, знайти певну відповідність даним. Але це вірно лише до певного моменту. Використання занадто великої кількості даних може призвести до того, що класифікатор стане дуже повільним під час побудови моделі, оскільки потрібно багато оптимізацій, щоб зробити модель загальною [4].

Оптимальний розмір набору даних, який використовується для побудови моделі класифікатора, залежить від ряду речей, таких як розмір проблеми класифікації, використовуваний алгоритм класифікатора та якість набору даних.

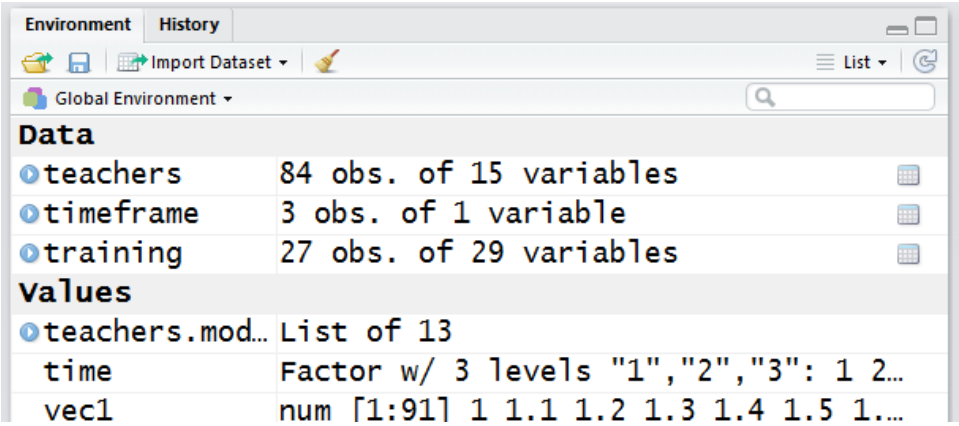
2.3. Опис програмного середовища

Є кілька типів інтерфейсу (пакетів або надбудов) з R Programming. Деякі поширені інтерфейси – це базовий R GUI, R Commander (пакет «Rcmdr», який використовується поверх основного R GUI) і RStudio. Опишемо переваги використання останнього [7].

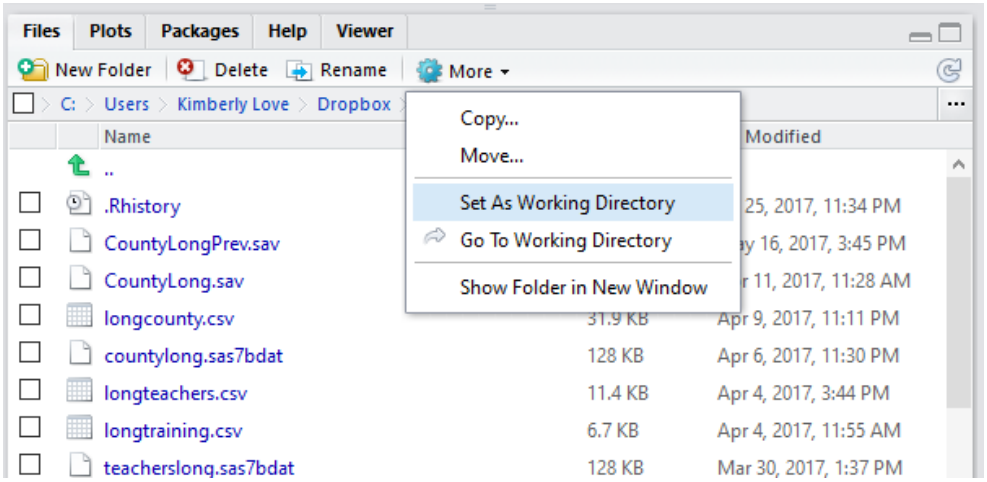
1) *RStudio розроблено, щоб полегшити написання коду.* Щойно користувач створює новий сценарій, вікна у його сеансі RStudio автоматично налаштовуються, щоб він міг бачити як свій сценарій, так і результати на консолі під час запуску свого синтаксису. Ще кращою є можливість викликати потенційні параметри синтаксису під час написання просто за допомогою клавіші табуляції. Наприклад, припустимо, що користувач намагається отримати доступ до змінної в наборі даних під назвою «вчителі», але я не запам'ятав назви змінних:



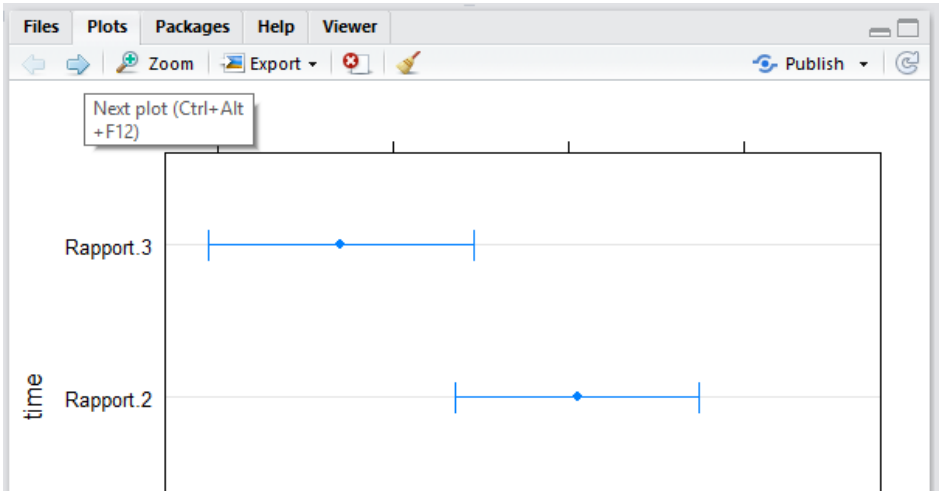
2) *RStudio дозволяє зручно переглядати та взаємодіяти з об'єктами, що зберігаються у середовищі користувача.* У базовому графічному інтерфейсі R користувач завжди зможе перерахувати об'єкти, які зберігаються у його середовищі. Але в RStudio є дуже корисне вікно «Середовище». Це показує всі об'єкти, які зберіг користувач, включаючи дані; скаляри, вектори та матриці; вихідні дані моделі; і т.д., а також короткий виклад інформації, яка зберігається в цих об'єктах. Користувач навіть може натиснути на свої набори даних безпосередньо, щоб відкрити їх і переглянути як електронні таблиці.



3) *RStudio дозволяє легко налаштувати робочий каталог і отримати доступ до файлів на комп'ютері*. Особливо, при роботі в Windows, однією з найбільш виснажливих частин програмування в R є налаштування робочого каталогу для доступу до необхідних файлів. За допомогою RStudio користувач має змогу переходити до папок на комп'ютері у вікні «Файли», переглядати будь-які файли, які є у цій папці, і встановлювати цю папку як робочий каталог.



4) *RStudio робить графіку набагато доступнішою для звичайного користувача.* Базовий R GUI вимагає від користувача певних зусиль, щоб зберегти графіку під час роботи. Але в RStudio є вікно, яке робить саме це. Користувач можете легко клацати вперед і назад між діаграмами, змінювати розміри ділянки без повторного запуску коду, а також експортувати або копіювати ділянки для включення в інші документи. Хоча він не має такої гнучкості, як графічні пристрої R, він надає все, що потрібно багатьом користувачам. Крім того, RStudio дозволяє науковцю використовувати графічні пристрої R будь-яким способом, яким він може використовувати їх у базовому графічному інтерфейсі R, якщо відчуває потребу (тобто користувач не обмежений використанням вікна графіків).



РОЗДІЛ ІІІ. МЕТОДИ КЛАСИФІКАЦІЇ ДАНИХ

У даному розділі розглянуто теоретичні підходи до методів класифікації даних, які використані у даній кваліфікаційній роботі. Кожен класифікатор описано в окремому підпункті даного розділу. До найпопулярніших алгоритмів класифікації даних відносяться такі алгоритми: Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Tree, Support Vector Machines, Random Forest.

3.1. Опис алгоритму Random Forest

Метод ансамблевого навчання для класифікації, регресії та інших завдань, який працює шляхом побудови безлічі дерев рішень під час навчання [12].

Алгоритм навчання для Random Forest застосовує загальну техніку завантажувального агрегування або пакетування до тих, хто вивчає дерева. Враховуючи навчальний набір $X = x_1, \dots, x_n$ з відповідями $Y = y_1, \dots, y_n$, пакетування повторно (B times) вибирає випадкову вибірку із заміною навчального набору та підбирає дерева до цих зразків:

для $b = 1, \dots, B$:

1. *Зразок, із заміною, n навчальних прикладів з X, Y; назвемо їх X_b, Y_b.*
2. *Класифікація або дерево регресії f_b, на X_b, Y_b.*

Після навчання прогнози для невидимих зразків x' можна зробити x' шляхом усереднення прогнозів усіх окремих дерев регресії на x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

або більшістю голосів у випадку класифікаційних дерев.

Ця процедура початкового завантаження призводить до кращої продуктивності моделі, оскільки вона зменшує дисперсію моделі, не збільшуючи зміщення.

Крім того, оцінку невизначеності прогнозу можна зробити як стандартне відхилення прогнозів від усіх окремих дерев регресії на x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

3.2. Ідея підходу *K*-найближчих сусідів

Метод використовується для класифікації та регресії. В обох випадках вхідні дані складаються з k найближчих навчальних прикладів у наборі даних [30].

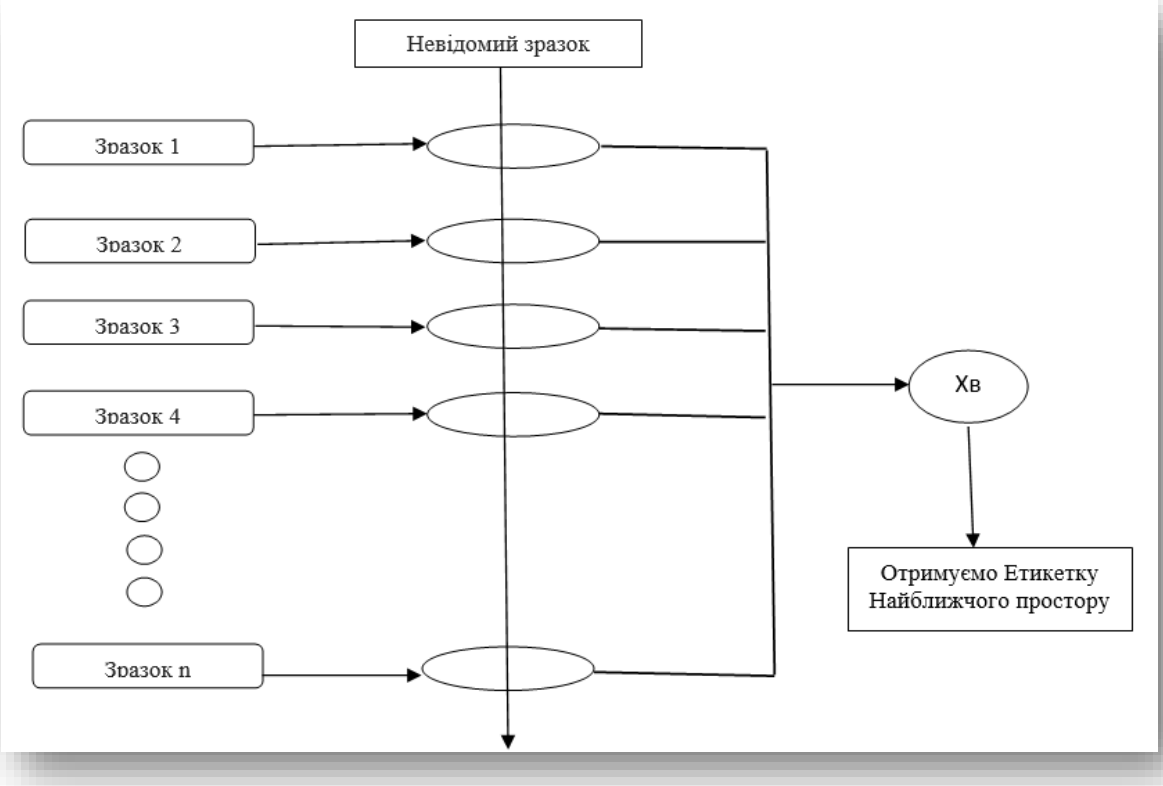


Рисунок 7. *K*-nearest neighbour

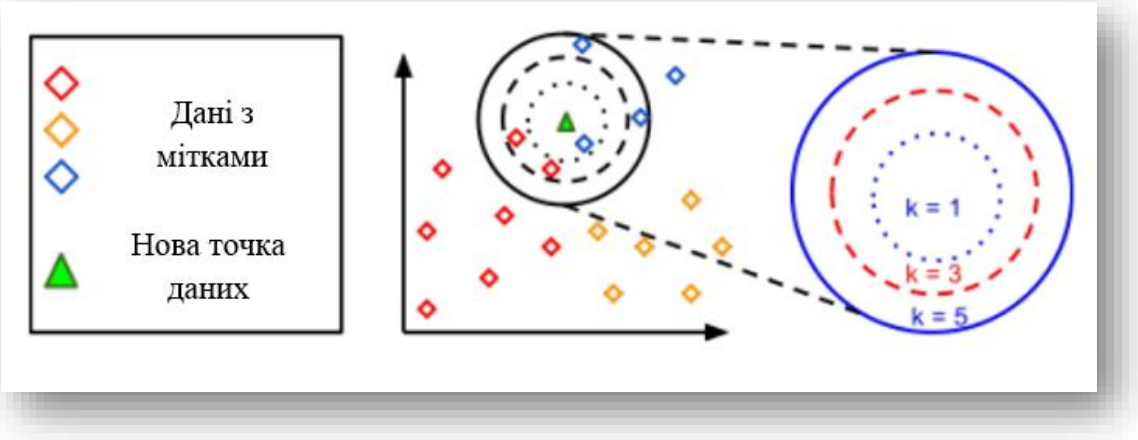


Рисунок 8. Процес зміни центрів кластерів

На попередньому рисунку показано процес переміщення центрів кластерів у просторі та класифікацію точок перед стабілізацією та поверненням знайдених кластерів.

K-nearest neighbour можна розглядати як присвоєння ваги k найближчим сусідам $1/k$ і всім іншим 0 вагу. Це можна узагальнити до зважених класифікаторів найближчих сусідів. Де i-му найближчому сусіду призначається вага w_{ni}, з $\sum_{i=1}^{n} w_{ni} = 1$. Аналогічний результат щодо сильної узгодженості зважених класифікаторів найближчих сусідів також має місце.

Позначимо C_n^{wnn} – зважений найближчий класифікатор з вагами $\{w_{ni}\}_{i=1}^{n}$. Цей класифікатор підлягає умовам регулярності, які в асимптотичній теорії є умовними змінними, які вимагають припущень для розрізнення параметрів за певними критеріями. На класових розподілах надлишковий ризик має наступне асимптотичне розкладання:

$$R_R(C_n^{wnn}) - R_R(C^{Bayes}) = (B_1 S_n^2 + B_2 t_n^2)\{1 + o(1)\},$$

для констант B_1 і B_2, де

$$S_n^2 = \sum_{i=1}^{n} w_{ni}^2 \text{ і } t_n = n^{-\frac{2}{d}} \sum_{i=1}^{n} w_{ni} \left\{ i^{1+\frac{2}{d}} - (i-1)^{1+\frac{2}{d}} \right\}.$$

Оптимальна схема зважування $\{w_{ni}^*\}_{i=1}^{n}$, що врівноважує два терміни, надається таким чином: встановити k^*

$$k^* = \left\lfloor B_n^{\frac{4}{d+4}} \right\rfloor,$$

$$w_{ni}^* = \frac{1}{k^*}\left[1 + \frac{d}{2} - \frac{d}{2k^{*2/d}}\left\{i^{1+2/d} - (i-1)^{1+2/d}\right\}\right]$$

для $i = 1,2,\dots,k^*$ та

$$w_{ni}^* = 0$$

$$i = k^* + 1, \dots, n.$$

З оптимальними вагами домінуючим членом в асимптотичному розкладі надлишкового ризику є $\mathcal{O}(n^{-\frac{4}{d+4}})$.

3.3. Переваги методу Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) – це методи класифікації та зменшення розмірності, які можна інтерпретувати з двох точок зору [31]:

- ймовірнісна інтерпретація
- тлумачення Фішера

Перша інтерпретація корисна для розуміння припущень LDA. Друга інтерпретація дозволяє краще зрозуміти, як LDA виконує зменшення розмірності.

- ***Імовірнісна інтерпретація.*** Кожен клас $k \in \{1, \dots, K\}$ присвоюється попередній $\widetilde{\pi k}$ так, що $\sum_{i=1}^{k} \widetilde{\pi k} = 1$. Згідно з правилом Байєса, апостеріорна ймовірність

  $$Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{1=1}^{K} f_1(x)\pi_1},$$

  де $f_k(x)$ є щільністю X обумовленою k. Максимально-апостеріорна (MAP) оцінка спрощується до

  $$G(x) = \arg\max_k Pr(G = k|X = x) = \arg\max_k f_k(x)\,\pi_k,$$

  тому що знаменник однаковий для всіх класів.

LDA припускає, що щільність є Гаусовою:

$$f_k(x) = |2\pi\sum_k|^{-1/2} \exp\left(-\frac{1}{2}(\mathrm{x} - \mu_k)^T \sum_k^{-1} (x - \mu_k)\right),$$

де $\sum_k$ – коваріаційна матриця для вибірок з класу k та $|.|$ є визначальною. LDA передбачає, що всі класи мають однакову коваріаційну матрицю, тобто $\sum_k = \sum, \forall k$.

- ***Інтерпретація Фішера.*** Критерій оптимізації LDA Фішера стверджує, що центроїди груп повинні бути рознесені якомога далі. Це означає знаходження лінійної комбінації $Z = a^T X$ такої, що a^T максимізує дисперсію між класами відносно дисперсії всередині класу. Дисперсія всередині класу дорівнює W, це об'єднана коваріаційна матриця $\widehat{\sum}$, що вказує на відхилення всіх спостережень від центроїдів їхнього класу. Міжкласова дисперсія визначається відповідно до відхилення центроїдів від загального середнього. Для Z, дисперсія між класами становить $a^T Ba$, а дисперсія всередині класу становить $a^T Wa$.

  Таким чином, LDA можна оптимізувати за допомогою коефіцієнта Релея

$$\max_a \frac{a^T Ba}{a^T Wa},$$

який визначає оптимальне відображення X до нового простору Z. $Z \in \mathbb{R}^{1\times p}$ тобто, спостереження відображаються в одному вимірі.

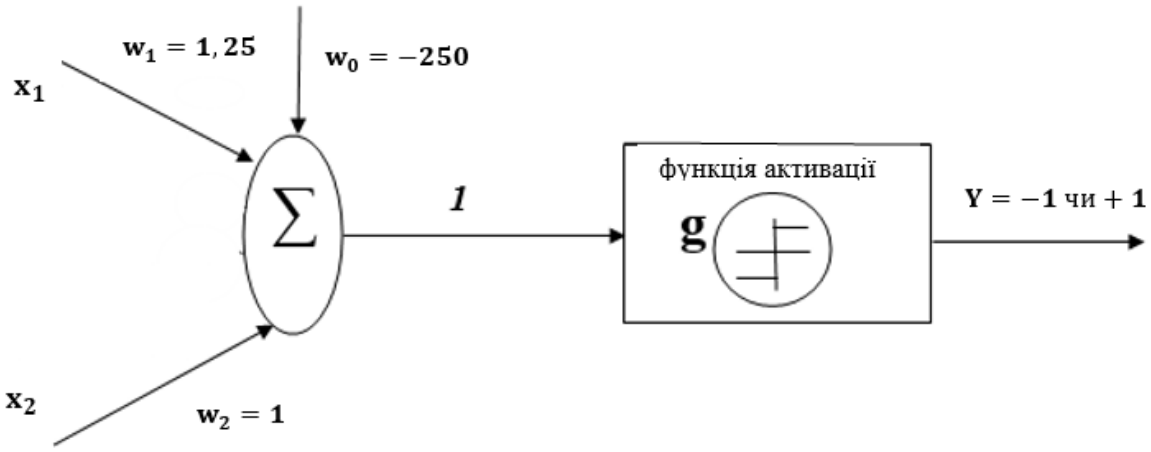


Рисунок 9. Моделювання лінійного класифікатора

3.4. Застосування Quadratic discriminant analysis

Quadratic Discriminant Analysis (QDA) – це варіант LDA, у якому індивідуальна коваріаційна матриця оцінюється для кожного класу спостережень [31]. QDA – особливо корисний, якщо є попередні знання про те, що окремі класи демонструють чіткі коваріації. Недоліком QDA є те, що його не можна використовувати як техніку зменшення розмірності.

У QDA потрібно оцінити $\sum_k$ для кожного класу $k \in \{1, \dots, K\}$, а не припускати $\sum_k = \sum$, як в LDA. Дискримінантна функція LDA є квадратичною до x:

$$\delta_k(x) = -\frac{1}{2}\log|\sum_k| - \frac{1}{2}(x - \mu_k)^T \sum_k^{-1} (x - \mu_k) + \log \pi_k.$$

Оскільки QDA оцінює коваріаційну матрицю для кожного класу, вона має більшу кількість ефективних параметрів, ніж LDA. Ми можемо отримати кількість параметрів наступним чином.

Потрібен K напередодні класу π_k. Оскільки $\sum_{i=1}^{K} \pi_k = 1$, то не потрібен параметр для одного з пріоритетів. Отже, $K - 1$ безкоштовні параметри для пріорів.

Оскільки є K центроїди, μ_k з p записами кожного є K_p – параметри, що стосуються засобів. З коваріаційної матриці $\sum_k$ потрібно розглянути лише діагональ і верхній правий трикутник. Ця область коваріаційної матриці має $\frac{p(p+1)}{2}$ елементів. Оскільки K таких матриць потрібно оцінити, є $K\frac{p(p+1)}{2}$ параметри, що стосуються коваріаційних матриць.

Таким чином, ефективна кількість параметрів QDA становить

$$K - 1 + K_p + K\frac{p(p+1)}{2}.$$

Оскільки кількість параметрів QDA є квадратичною в p, QDA слід використовувати обережно, якщо простір функцій великий.

Рис. 10 показує межі прийняття рішень для LDA та QDA. Нижній рядок демонструє, що LDA може застосовувати лише лінійні межі, тоді як QDA може застосовувати квадратичні межі, тому є більш гнучким.

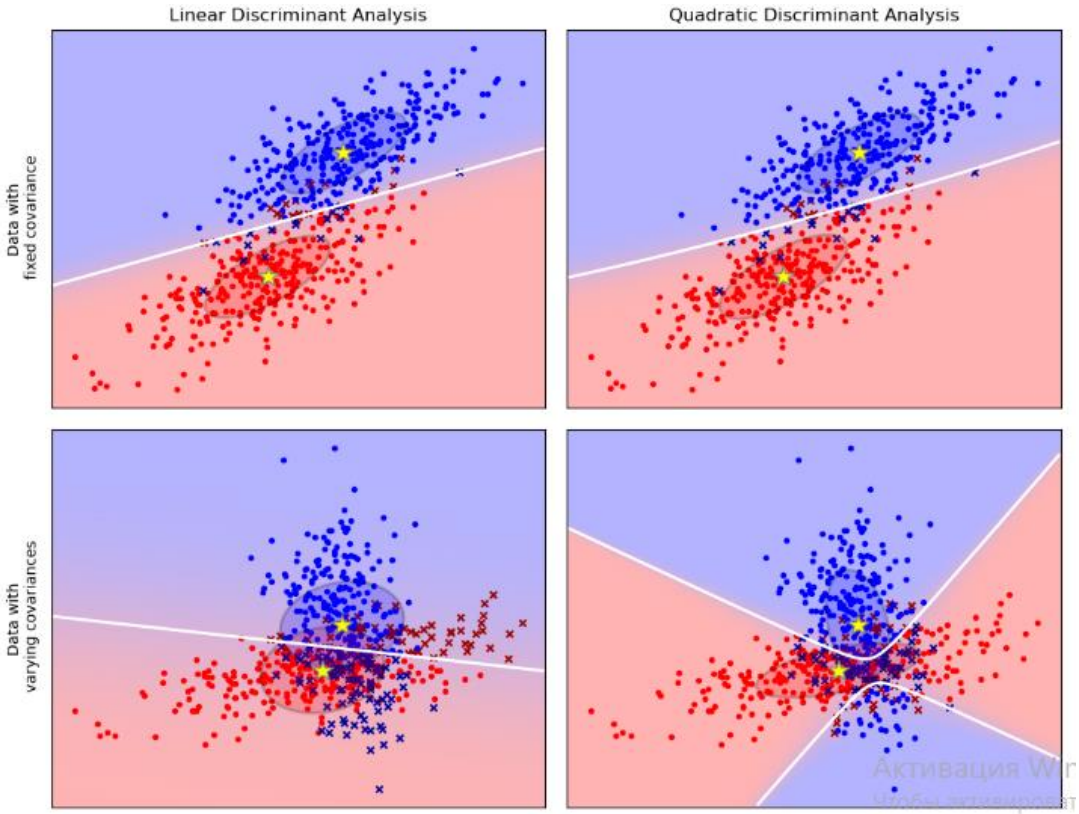


Рисунок 10. Лінійні та квадратичні межі класифікації методів LDA та QDA

3.5. Алгоритм Support Vector Machine

Support Vector Machine (SVM) – це набір пов’язаного навчання під наглядом методів, що використовуються для класифікації та регресії [30].

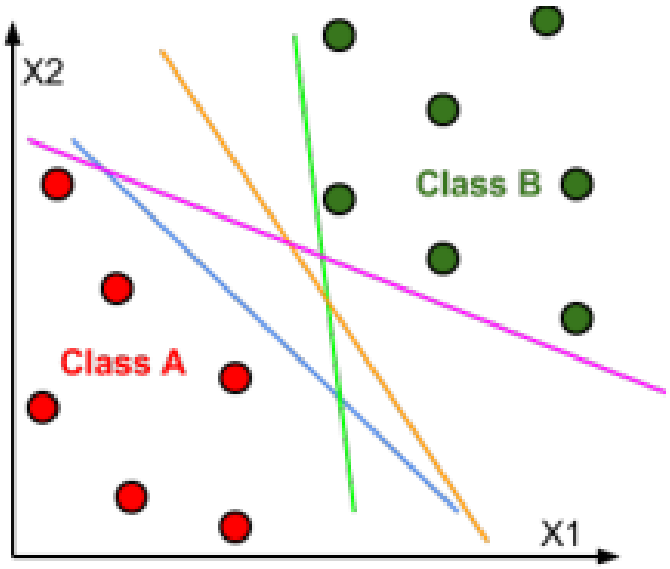


Рисунок 11. Класифікація даних 4-ма гіперплощинами

На Рис. 11 зображено набір точок із тестових даних, які можна розділити лінійно. Дані точки були розділені 4 гіперплощинами, які класифікують їх правильно.

Властивістю SVM є одночасне зведення до мінімуму емпіричної помилка класифікації та максимізації геометричного запасу. Точки даних форми

$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4) \dots \dots \dots, (x_n, y_n)\},$$

де y_n = 1/−1, константа, що позначає клас якій ця точка x_n належить. n – число-зразок. Кожен x_n є вимірним дійсним вектором. Масштабування важливо захистити від змінних (атрибутів) з більшою варіацією. Можливо розглянути масштабування за допомогою навчальних даних (training data), за допомогою ділення (або seperating) гіперплощини, яка приймає

$$w \cdot x + b = 0,$$

де b – скаляр, а w – p-вимірний вектор.

Паралельні гіперплощини можуть описуватися рівняннями

$$w \cdot x + b = 1,$$

$$w \cdot x + b = -1.$$

Якщо навчальні дані є лінійно розподіленими, то можна вибрати ці гіперплощини так, щоб не було точок між ними, а потім спробувати максимізувати їх відстань. За допомогою геометрії знаходимо відстань між гіперплощиною $\frac{2}{|w|}$, мінімізуємо $|w|$. Потрібно переконатися, що для всіх i також

$$w \cdot x_i - b \geq 1 \text{ або } w \cdot x_i - b \leq -1.$$

Це можна записати наступним чином

$$y_i(w \cdot x_i - b) \geq 1, \quad 1 \leq i \leq n.$$

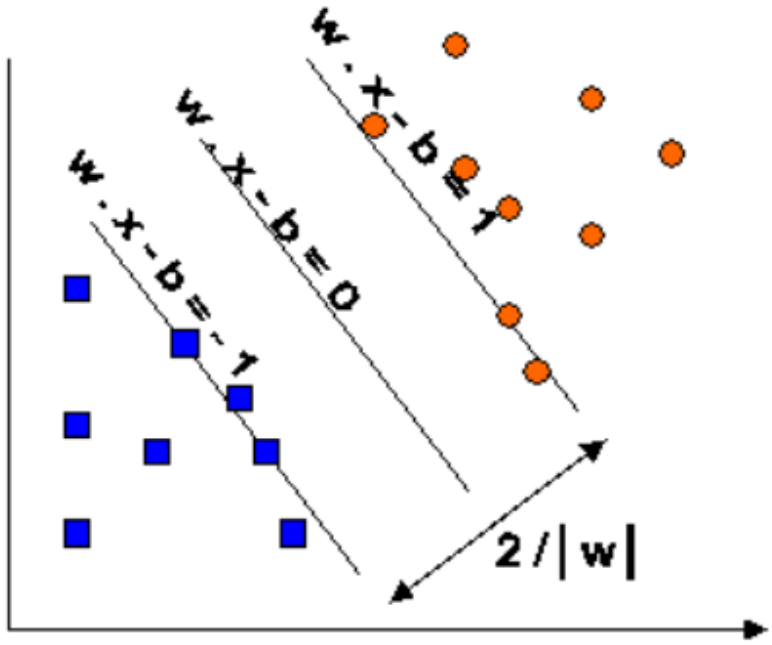


Рисунок 12. Гіперплощини максимального поля для SVM із зразками з двох класів

РОЗДІЛ IV. КЛАСИФІКАЦІЯ ДАНИХ У СЕРЕДОВИЩІ R

4.1. Вхідні дані

Набір даних, використаний для класифікації у даному дослідженні, містить інформацію про успішність учнів середньої школи з математики, включаючи їхні оцінки та демографічну інформацію. Дані були зібрані з трьох середніх шкіл у Сполучених Штатах. Набір даних доступний за посиланням http://roycekimmons.com/tools/generated_data/exams або може бути завантажений з платформи Kaggle https://www.kaggle.com/datasets/rkiattisak/student-performance-in-mathematics.

Вхідні дані завантажено у форматі .csv (1000 записів, 8 стовпців). У наступній таблиці наведено опис змінних, які присутні в даних

№	Назва змінної у наборі даних	Тлумачення
1.	Gender	стать студента (чоловік/жінка)
2.	Race/ethnicity	расове чи етнічне походження студента (азіат, афроамериканець, латиноамериканець тощо)
3.	Parental level of education	найвищий рівень освіти, отриманий батьками або опікунами учня
4.	Lunch	чи отримує студент безкоштовний обід або обід за зниженою ціною (так/ні)
5.	Test preparation course	чи завершив студент курс підготовки до тесту (так/ні)
6.	Math score	оцінка студента за стандартизований тест з математики
7.	Reading score	Оцінка студента за стандартизованим тестом читання
8.	Writing score	Оцінка студента за стандартизованим тестом з письма

Основна увага у даній роботі присвячена класифікації даних. Тому змінна відповіді повинна мати бінарний тип. Таким чином змінну Test preparation course для початку потрібно перекодувати у «1» - так та «0» - ні. Незалежними змінними

у даній роботі виступатимуть змінні Math score та Reading score. Іншими словами, за оцінками з математики та письма учнів можна розділити на дві групи, де до першої групи належать ті, хто завершив підготовку до тестів, а до другої – ті, хто не проходив курси. Суть даного дослідження полягає у статистичних підходах та алгоритмах класифікації. Дані можуть бути вибраними з будь-якої сфери життєдіяльності людини. Тому навіть згенеровані дані з допомогою процесу Монте-Карло можуть бути використаними для демонстрації алгоритмів класифікації. Основна вимога полягає у тому, щоб незалежні змінні були числового типу, а змінна відповіді – бінарного.

Описові статистики вхідних даних:

- NA values відсутні в даних.
- Обидві незалежні змінні X1_1 (оцінка з математики) та X2 (оцінка з письма) є дійсного типу, залежна змінна Group – факторного типу.
- Вхідні дані складаються з 1000 спостережень для кожної змінної.

Описові статистики Group = 0, тобто тих учнів, які не проходили курс підготовки до тесту:

- Змінна X1_1 змінюється від 15 до 100 балів. Отже, студенти, які не проходили курс підготовки до тесту, таких можуть успішно його пройти з математики на максимальну кількість балів. Середнє значення для змінної X1_1 становить 66,5 балів.
- Змінна X2 змінюється від 25 до 100 балів з середнім значенням 68,1. Логічно, що учень, який не проходив підготовку з математики та письма, більше балів отримає за тест з письма. Як бачимо і в цьому випадку без підготовки можна отримати 100 балів.
- Обидві змінні – оцінка з математики та письма мають розподіл, близький до нормального розподілу.

Описові статистики Group = 1, тобто тих учнів, які проходили курс підготовки до тесту:

- Змінна X1_1 змінюється від 27 до 100 балів. Середнє значення для змінної X1_1 становить 70,3 балів. Як бачимо середнє значення оцінки з математики за тест зросло майже на 4 бали внаслідок проходження курсу підготовки.
- Змінна X2 змінюється від 38 до 100 балів з середнім значенням 74,73. Середній показник кількості балів за письмо зріс майже на 7 балів.
- Обидві змінні – оцінка з математики та письма мають розподіл, близький до нормального розподілу.

На наступному рисунку наведено діаграму розсіювання даних результатів тесту з математики (X1_1) та письма (X2). Як видно із наступної діаграми, студенти, які успішно складають іспит з математики, також успішно здають і іспит з письма, що відображає реальну ситуацію (на прикладі нашого українського ЗНО). Тобто вказані стовпці даних є сильно корельованими ($\boldsymbol{correlation = 81.18\%}$).

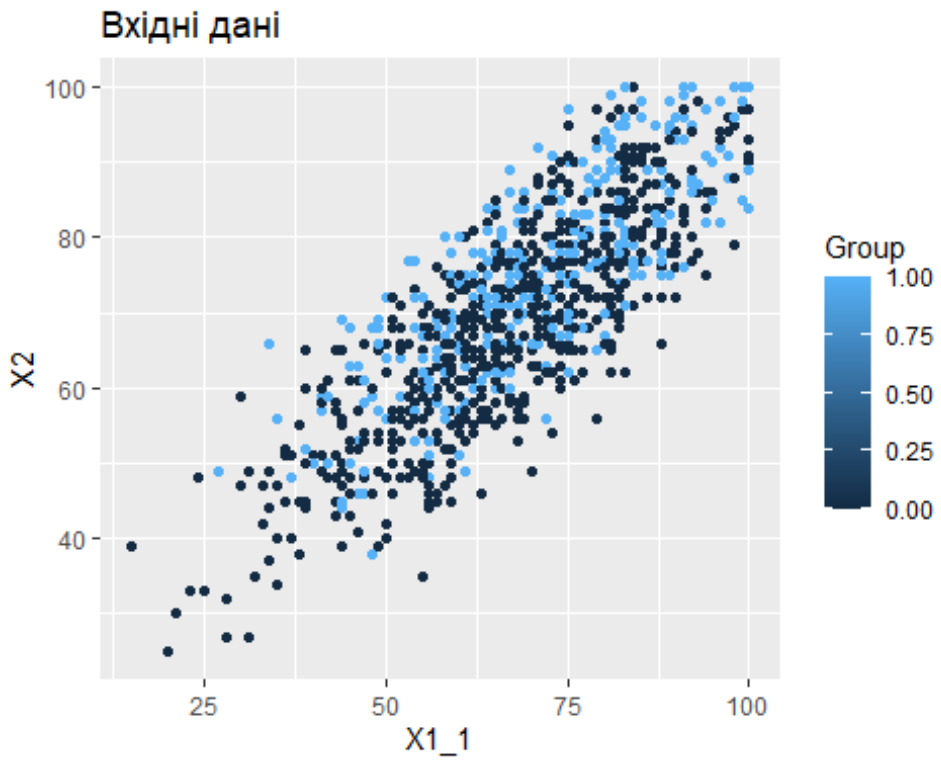


Рисунок 13. Діаграма розсіювання вхідних даних

На початку аналізу дані розділено на навчальну (80% вхідних даних) та тестову (20% вхідних даних) вибірки. Кожна із моделей класифікаторів будується на навчальному наборі, а далі – тестується на тестовому наборі дані. Наступним кроком є порівняння реальних тестових даних з прогнозованими даними, отриманими з допомогою моделі. Таким чином відбувається обчислення акуратності роботи моделі кожного класифікатора. Акуратність отриманої моделі якраз і показує відсоток спів падіння – скільки даних дійсно правильно є класифікованими. Далі розглянемо аналіз результатів кожної моделі класифікаторів окремо та здійснимо порівняння отриманих результатів.

4.2. Random Forest

Отримана точність класифікації на навчальній вибірці становить 66,83%. Кількість дерев – 500. Акуратність моделі Random Forest для тестових даних – 69,88%, що свідчить про те, що не всі значення класифіковані правильно. Відсоткове співвідношення неправильно класифікованих даних приблизно становить становить 30%.

Зниження кількості неправильно класифікованих даних, яке ми спостерігаємо, пов’язане з ідіосинкразичним компонентом даних. Рішення будь-якого дерева є певним типом випадкової змінної, і очікувана частота помилок для будь-якої кількості дерев $\boldsymbol{T}$ є виключно особливістю $\boldsymbol{\varepsilon_i}$, яка буде різною для кожного набору даних. Збільшення кількості дерев спричиняє коливання частоти помилок.

4.3. Support Vector Machine

SVM є одним із найпопулярніших алгоритмів керованого навчання, який використовується для задач класифікації та регресії. Однак, в основному, SVM є широко застосовним для проблем класифікації в машинному навчанні.

Метою алгоритму SVM є створення найкращої лінії або межі рішення, яка може розділити $\boldsymbol{n}$-вимірний простір на класи, щоб ми могли легко помістити нову точку даних у правильну категорію в майбутньому. Така оптимальна лінія рівня називається гіперплощиною.

SVM вибирає крайні точки/вектори, які допомагають створити гіперплощину. Крайні вектори називають опорними векторами, а отже, алгоритм називають машиною опорних векторів. Для тренувальної вибірки при використанні радіального типу ядра (kernel = «radial») точність класифікації даних становить 66,71%, а для тестової – 64,14%. Тип ядра – це функція, яка розділяє точки даних у просторі.

```
## Confusion Matrix and Statistics
##
##   p1
##     0   1
##  0 501  22
##  1 245  34
##
##            Accuracy : 0.6671
##              95% CI : (0.6333, 0.6997)
##    No Information Rate : 0.9302
##    P-Value [Acc > NIR] : 1
##
##               Kappa : 0.0981
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.6716
##          Specificity : 0.6071
##       Pos Pred Value : 0.9579
##       Neg Pred Value : 0.1219
##           Prevalence : 0.9302
##       Detection Rate : 0.6247
##   Detection Prevalence : 0.6521
##     Balanced Accuracy : 0.6394
```

```
##
##       'Positive' Class : 0
```

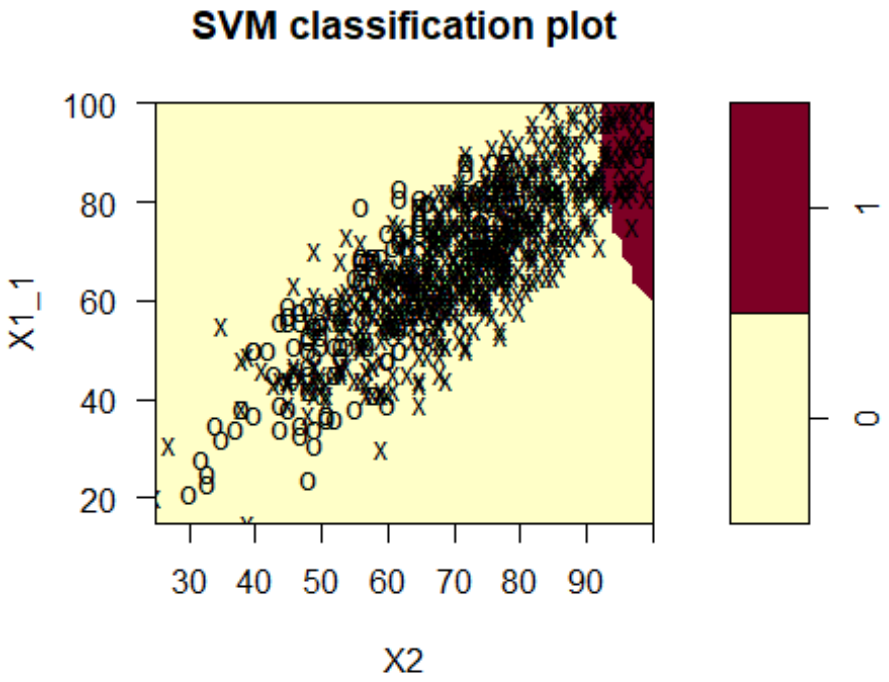


Рисунок 14. Класифікатор SVM для навчальної вибірки (radial kernel)

При використанні поліноміального ядра (kernel = «polynomial») точність класифікації даних збільшено до 66,83%, а для тестової – 66,67%.

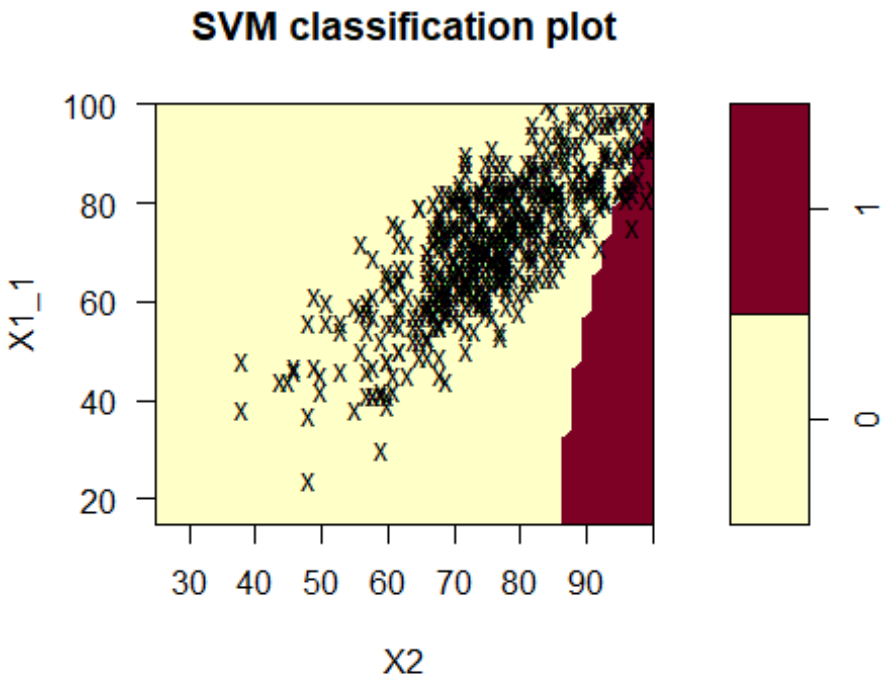


Рисунок 15. Класифікатор SVM для навчальної вибірки (polynomial kernel)

При використанні kernel = “sigmoid” точність падає нижче 60%. Очевидно, що в даному випадку використовувати лінійну СВМ недоцільно.

4.4. *K* nearest neighbours

K Nearest Neighbors – це базовий алгоритм, який зберігає всі доступні маркери даних та передбачає класифікацію немаркованих даних на основі показника подібності. У лінійній геометрії, коли два параметри відображаються в двовимірній декартовій системі, ми визначаємо міру подібності, обчислюючи відстань між точками. Те саме стосується і тут, алгоритм ***KNN*** працює на основі припущення, що подібні речі існують поблизу.

Щоб уникнути надмірного чи недостатнього підгонки нашої моделі та знайти баланс між зміщенням і дисперсією моделі, ми розглядаємо використання параметра налаштування для вибору оптимального гіперпараметра. Ми використали підхід ***k***-кратної перехресної перевірки, щоб розділити дані нашого

навчального набору на 10 вибірок і 3 рази ітеративно провести навчання зі значенням K. У кожному циклі 9 різних наборів з 10 використовуються для навчання, а 1 набір використовується для перехресної перевірки. Цей процес оптимізації показав, що 60 є оптимальним $\boldsymbol{K}$-значенням і був використаний для моделювання $\boldsymbol{KNN}$.

Крива ROC (receiver operating characteristic curve) – це графік, що показує продуктивність моделі класифікації при переборі порогів класифікації. Ця крива відображає два параметри: True Positive Rate та False Positive Rate.

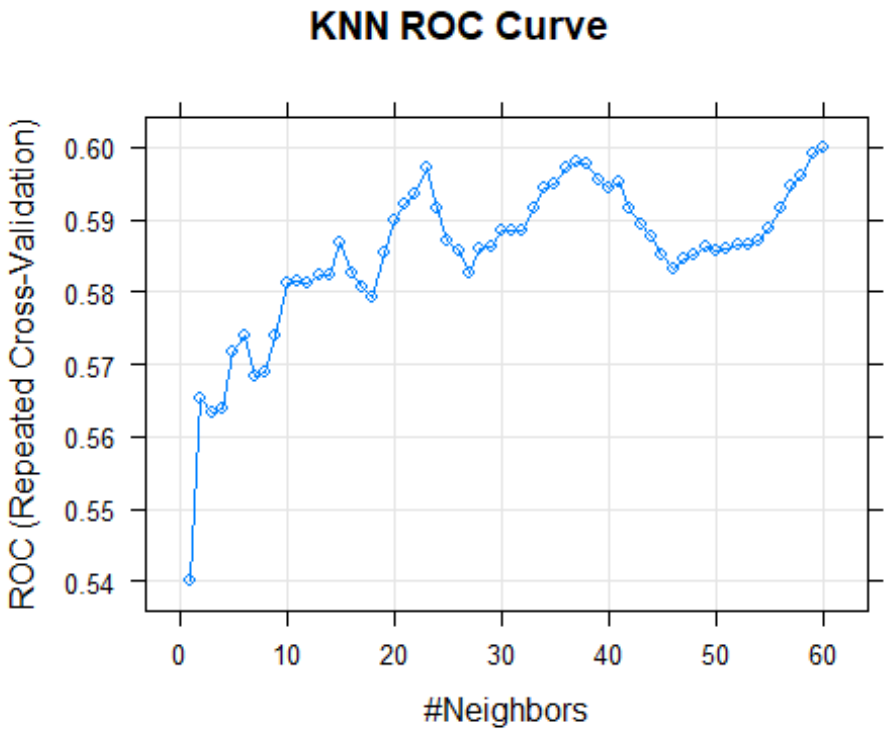


Рисунок 16. Крива $\boldsymbol{ROC}$ для алгоритму $\boldsymbol{KNN}$

$\boldsymbol{KNN}$ модель має 7,2% частоту помилок навчання. Для тестового набору даних показник точності моделі становить 91,35%, що є прекрасним показником точності класифікації.

4.5. Linear discriminant analysis

Лінійний дискримінантний аналіз (LDA) є узагальненням лінійного дискримінантного аналізу Фішера, методу, який використовується в статистиці та інших галузях для пошуку лінійної комбінації ознак, які характеризують або розділяють два або більше класів. LDA є одним із найпопулярніших методів зменшення розмірності, який використовується для задач контрольованої класифікації в машинному навчанні.

Вхідні дані сильно перекриваються, тому алгоритм LDA досягає лише 66,54% точності на навчальному наборі даних. Точність класифікації методом LDA на тестовій вибірці становить 69,71%.

4.6. Quadratic discriminant analysis

Лінійний дискримінантний аналіз (Linear Discriminant Analysis) і квадратичний дискримінантний аналіз (Quadratic Discriminant Analysis) є двома класичними класифікаторами з лінійною та квадратичною поверхнею прийняття рішень, відповідно. Ці класифікатори зручні у використанні, тому що вони мають закриті рішення, які можна легко обчислити; за своєю суттю є багатокласовими; добре працюють на практиці та не мають гіперпараметрів для налаштування.

У нашому випадку точність класифікації методом LDA для навчальної вибірки становить 66,67%, а для тестової вибірки – 68,75%. Як бачимо, класифікатор QDA для розглянутих даних у роботі показує навіть гірший результат класифікації, ніж LDA.

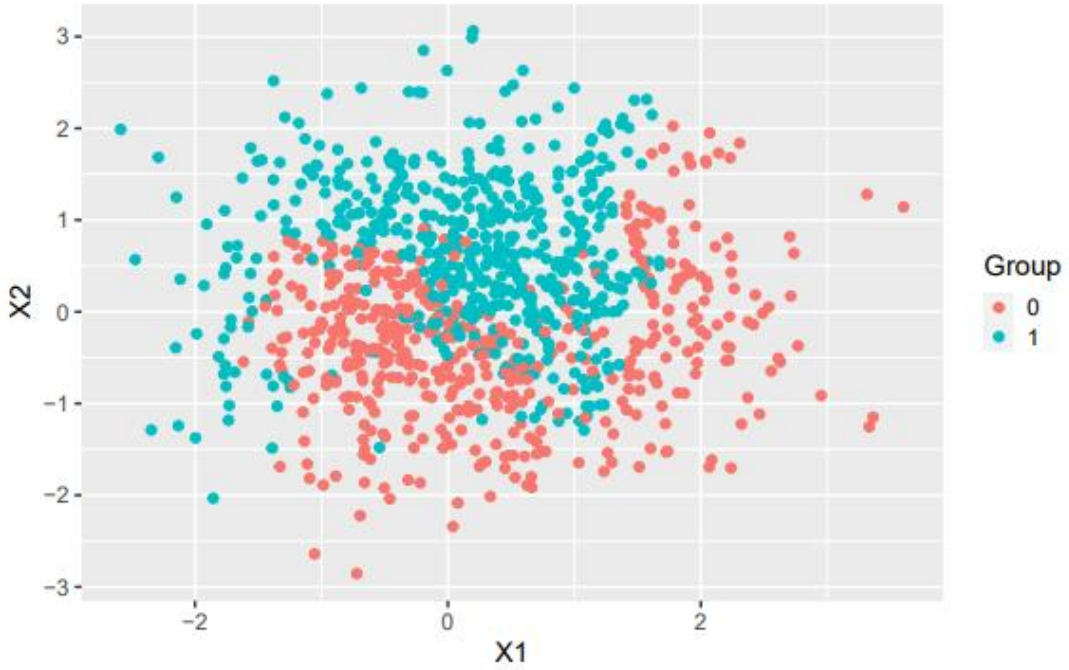
4.7. Порівняння результатів класифікаторів

Як бачимо усі розглянуті алгоритми класифікації показують точність класифікації на тестовій вибірці приблизно від 66% до 70%.

Таблиця 1. Результати класифікації даних, які сильно перекриваються

№	Класифікатор	Навчальна вибірка	Тестова вибірка	Sensitivity	Specificity
1.	Random Forest	66,83%	69,88%	67%	60%
2.	SVM	66,83%	66,67%	66%	20%
3.	KNN	66,67%	64,42%	66%	58%
4.	LDA	66,54%	69,71%	-	-
5.	QDA	66,67%	68,75%	-	-

Розглянемо дані, які не так сильно перекриваються, як у попередньому прикладі. Дані на Рис. 17 є результатом симуляції методом Монте-Карло. На перший погляд, у цьому випадку групи можуть бути класифікованоми за допомогою моделей з вищою точністю, ніж у попередньому випадку. Перевіримо цю гіпотезу за допомогою п'яти тих же алгоритмів класифікації. Результати класифікації наведено у Таб. 2.



Таблиця 2. Результати класифікації даних, які НЕ сильно перекриваються

№	Класифікатор	Навчальна вибірка	Тестова вибірка	Sensitivity	Specificity
1.	Random Forest	100%	87%	85%	83%
2.	SVM	87%	88%	84%	95%
3.	KNN	91%	86%	87%	85%

4.	LDA	70%	76%	-	-
5.	QDA	73%	77%	-	-

Як бачимо, у даному випадку алгоритм RF досягає 100% точності на навчальній вибірці та 87% – на тестові. Стовідсоткова точність вказує на те, що модель RF є ідеальним класифікатором для цих вхідних даних. У загальному, результати класифікації такої вибірки є значно кращими. Серед розглянутих алгоритмів найвищу точність класифікації на тестовій вибірці досягає алгоритм SVM (88%), а найнижчу – LDA (76%). Очевидно, що LDA показав найгірший результат класифікації через нелінійну «межу» між вхідними даними.

Можемо підсумувати, що трійкою-лідерами серед найкращих алгоритмів класифікації є: Random Forest, SVM, KNN. При розгляді даних спеціаліст може використовувати різні метрики відстаней, враховувати або не враховувати кореляцію між даними – тим самим покращуючи якість класифікації даних.

Висновки

У даній кваліфікаційній роботі розглянуто 5 методів класифікації даних: Random Forests, SVM, KNN, LDA, QDA. Ефективність класифікації кожної із побудованих моделей визначено за допомогою оцінки акуратності моделі. Акуратність кожної моделі оцінюється шляхом порівняння прогнозованих класів (за допомогою навчальної вибірки) та реальних класів (за допомогою тестової вибірки). Ґрунтуючись на результати класифікації, можна зробити висновок, що підготовчий курс до іспитів має позитивний вплив на результати іспитів з математики та письма. Усі з розглянутих класифікаторів досягають точності класифікації на тестовій вибірці приблизно на однаковому рівні – від 66% до 70%. Кращим класифікатором для вибірки реальних даних є Random Forest, який досягає 69,88% точності на тестовій вибірці з чутливістю моделі 67%.

Результати класифікації симульованих даних за допомогою п’ятьох алгоритмів машинного навчання наведено в Таблиці 2. Чутливість та специфічність не обчислено для моделей LDA та QDA у зв’язку з тим, що ці алгоритми не забезпечують достатньої точності класифікації. Як бачимо, результати класифікації за методами KNN, SVM і RF мало відрізняються. Найкращу точність було отримано за допомогою методу SVM 88% з ядром = "радіальний". Незважаючи на те, що результати класифікаційної точності методу випадкового лісу лише на 1% нижче, все ж хочу віддати перемогу також RF-методу у випадку симульованих даних, оскільки значення чутливості становить 85%, що на 1% менше ніж чутливість SVM. Специфічність для RF на 12% менше, ніж SVM. Тому найкращим класифікатором для симульованих даних є випадковий ліс. Це пов’язано з тим, що метод випадкового лісу усуває недоліки алгоритму дерева рішень, зменшуючи переобладнання набору даних і підвищуючи точність.

Протягом роботи над кваліфікаційною роботою я навчився працювати з реальними даними – визначати аутлаєри, аналізувати описові статистики, будувати моделі класифікаторів – налаштовувати гіперпараметри, обчислювати точність отриманих моделей тощо. Дана кваліфікаційна робота ще раз підтверджує гнучкість алгоритму Random Forest у багатьох сферах застосування.

Список використаних джерел

[1] Ani, R., Sasi, G., Sankar, U.R., Deepa, O.S.: Decision support system for diagnosis and prediction of chronic renal failure using random subspace classification. In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1287-1292. Jaipur (2016).

[2] B. et al, Proceedings of the 5th Annual Workshop on Computational Learning Theory COLT'92 - A Training Algorithm for Optimal Margin Classifiers, pp. 144-152. ACM Press, 1 ed., 1992.

[3] Branwen, G.: Silk road: theory and practice (2015). http://www.gwern.net/SiIk%20Road, Accessed 09 Dec 2015.

[4] D. Aha and D. Kibler, *Machine Learning - Instance-based learning algorithms*, pp. 37-66. Kluwer Academic Publishers, 1 ed., 1991.

[5] Denoyer, L., Gallinari, P.: Bayesian network model for semi-structured document classification. Inf. Process. Manag. (Elsevier) 40(5), 807-827 (2004).

[6] Denoyer, L., Vittaut, J., Gallinari, P., Brunessaux, S., Brunessaux, S.: Structured multimedia document classification. In: ACM DOCENG 2003, pp. 153-160 (2003).

[7] Ding, Y., Korotkiy, M., Omelayenko, B., Kartseva, V., Zykov, V., Klein, M., Schulten, E., Fensel, D.: GoldenBullet: automated classification of product data in e-commerce. In: Business Information System (2002).

[8] E. Frank, "The weka classifier reptree." http://weka.sourceforge.net/doc.dev/ weka/classifiers/trees/REPTree.html. Accessed: 2012-12-30.

[9] *EMC*^{2}, "Digital universe." http://www.emc.com/leadership/programs/ digital-universe.htm. Accessed: 2012-09-25.

[10] Fensel, D., Ding, Y., Schulten, E., Omelayenko, B., Botquin, G., Brown, M., Flett, A.: Product data integeration in B2B e-commerce. IEEE Intell. Sys. 16(3), 54-59 (2001).

[11] G. H. John and P. Langley, Eleventh Conference on Uncertainty in Artificial

Intelligence - Estimating Continuous Distributions in Bayesian Classifiers, pp. 338-345. Morgan Kaufmann, 1 ed., 1995.

[12] J. Platt, “Advances in kernel methods, support vector learning - fast training of support vector machines using sequential minimal optimization.” http://research. microsoft.com/apps/pubs/?id=68391. Accessed: 2012-09-25.

[13] James Manyika et al for McKinsey Global Institute, “Big data: The next frontier for innovation, competition, and productivity.” http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_ frontier_for_innovation. Accessed: 2012-09-25.

[14] K.-B. D. et al, Multiple classifier systems : 6th international workshop - Which Is the Best Multiclass SVM Method? An Empirical Study, pp. 278-285. Berlin : Springer, 1 ed., 2005.

[15] M. A. et al, Automation and Remote Control - Theoretical foundations of the potential function method in pattern recognition learning, pp. 821-837. Springer, 1 ed., 1964.

[16] Mehanna, Fadi Samih Omar, "Towards a scalable and efficient data classification technique." (2005).

[17] Mitchell, T.: Machine Learning. McGraw-Hill, Columbus (1997).

[18] Priyanka, C., Gupta, D.: Fine grained sentiment classification of customer reviews using computational intelligent technique. Int. J. Eng. Technol. 7(4), 1453-1468 (2015).

[19] R. C. Holte, *Machine Learn Vol 11 - Very Simple Classification Rules Perform Well on Most Commonly Used Datasets*, pp. 63-90. Kluwer Academic Publishers- Plenum Publishers, 1 ed., 1993. Available at: http://dx.doi.org/10.1023/A: 1022631118932.

[20] R. Kohavi, 8th European Conference on Machine Learning - The Power of

Decision Tables, pp. 174-189. Springer, 1 ed., 1995.

[21] R. Quinlan, *Cf.5: Programs for Machine Learning*, pp. -. Morgan Kaufmann Publishers, 1993.

[22] Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. 34(1), 1-47 (2002).

[23] The Library of Congress, "The library of congress - web archiving faqs." **http: //**www.loc.gov/webarchiving/faq.html. Accessed: 2012-09-25.

[24] U. F. et al, *From Data Mining to Knowledge Discovery in Databases*, pp. 37-54. AI Magazine, 17 ed., 1991. 123.

[25] V. V. et al, Automation and Remote Control - Pattern recognition using generalized portrait method, p. 24. Springer, 1 ed., 1963.

[26] Vinithra, S.N., Anand Kumar, M, Soman, K.P.: Analysis of sentiment classification for Hindi movie reviews: a comparison of different classifiers. Int. J. Appl. Eng. Res. 10 (2015).

[27] W. W. Cohen, Twelfth International Conference on Machine Learning - Fast Effective Rule Induction, pp. 115-123. Morgan Kaufmann, 1 ed., 1995.

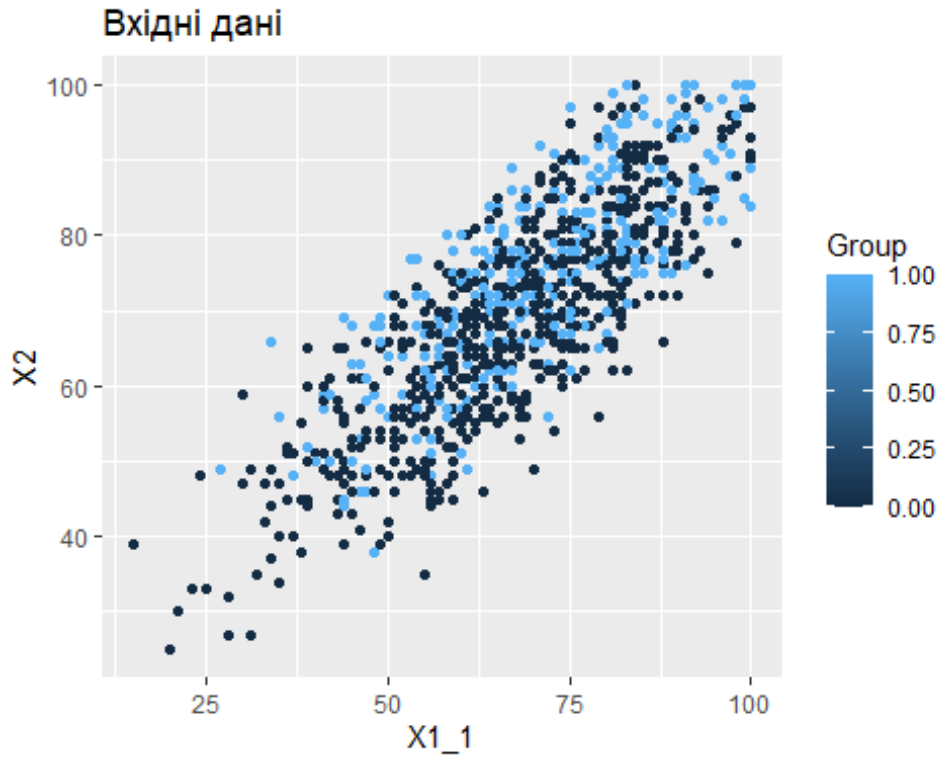
[28] Zaki, M.J., Aggarwal, C.C.: XRules: an effective structural classifier for XML data. In: 9th ACM SIGKDD, pp. 316-325 (2003).

[29] https://www.researchgate.net/publication/285663733_Data_classification_using_support_vector_machine

[30] https://www.datascienceblog.net/post/machine-learning/linear-discriminant-analysis/

Додатки

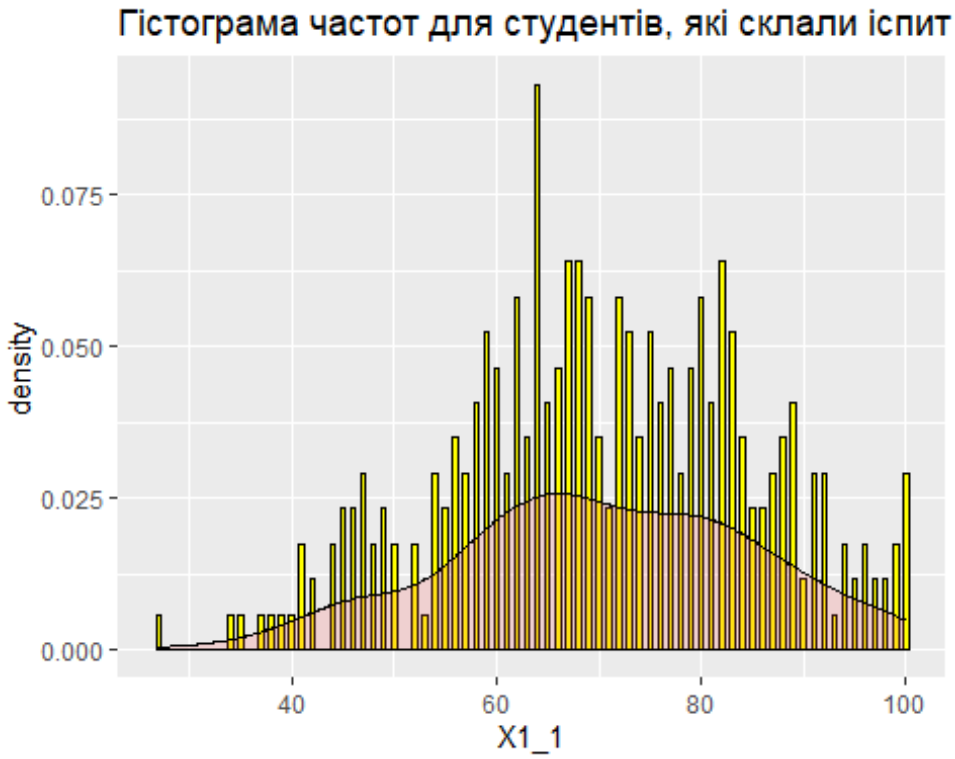
```
# Завантаження даних
df <- read_delim("exams.csv", delim = ";", escape_double = FALSE, trim_ws = TRUE
)
# Подивимося на кілька перших рядків даних
head(df)

## # A tibble: 6 × 8
##   gender `race/ethnicity` parental level…¹ lunch test …² math …³ readi…⁴ writi…⁵
##   <chr>  <chr>           <chr>            <chr> <chr>    <dbl>   <dbl>   <dbl>
## 1 female group D         some college     stan… comple…     59      70      78
## 2 male   group D         associate's deg… stan… none       96      93      87
## 3 female group D         some college     free… none       57      76      77
## 4 male   group B         some college     free… none       70      70      63
## 5 female group D         associate's deg… stan… none       83      85      86
## 6 male   group C         some high school stan… none       68      57      54
## # … with abbreviated variable names ¹`parental level of education`,
## #   ²`test preparation course`, ³`math score`, ⁴`reading score`,
## #   ⁵`writing score`

# Перевіряємо дані на пропущені значення
sum(is.na(df))

## [1] 0

# Variable Group should be a factor variable (0 and 1)
df$Group <- as.factor(df$`test preparation course`)
df <- df %>%
      mutate(Group = ifelse(Group == "none",0,1))


table(df$Group)

##
##   0   1
## 656 344

df$X1_1 <- df$`math score`
df$X2 <- df$`reading score`
# Basic scatter plot
ggplot(df, aes(x=X1_1, y=X2, color = Group)) + geom_point() + ggtitle("Вхідні да
ні")
```



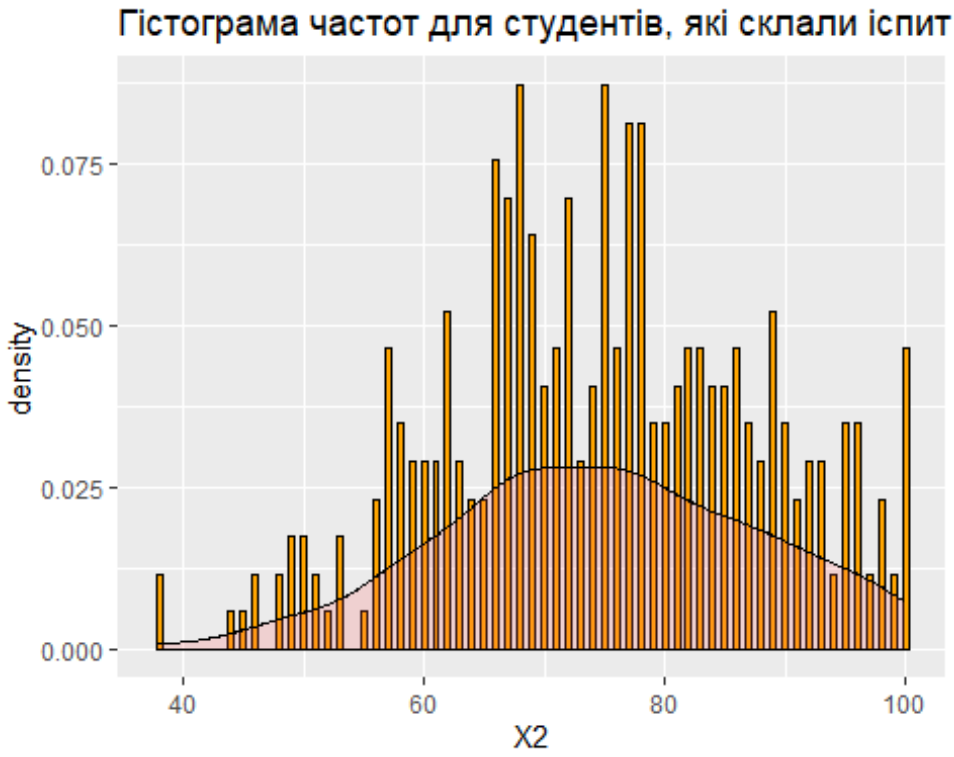

```
# Гістограма та щільність  розподілу
ggplot(df[df$Group == 0,], aes(x=X1_1)) +
  geom_histogram(aes(y=..density..),
                 binwidth=.5,
                 colour="black", fill="pink") +
  geom_density(alpha=.2, fill="#FF6666") + ggtitle("Гістограма частот для студен
тів, які не склали іспит з математики")
```


```
ggplot(df[df$Group == 0,], aes(x=X2)) +
  geom_histogram(aes(y=..density..),
                 binwidth=.5,
                 colour="black", fill="red") +
  geom_density(alpha=.2, fill="#FF6666") + ggtitle("Гістограма частот для студен
тів, які не склали іспит з письма")
```


```
ggplot(df[df$Group == 1,], aes(x=X1_1)) +
  geom_histogram(aes(y=..density..),
                 binwidth=.5,
                 colour="black", fill="yellow") +
  geom_density(alpha=.2, fill="#FF6666") + ggtitle("Гістограма частот для студен
тів, які склали іспит з математики")
```




```
ggplot(df[df$Group == 1,], aes(x=X2)) +
  geom_histogram(aes(y=..density..),
                 binwidth=.5,
                 colour="black", fill="orange") +
  geom_density(alpha=.2, fill="#FF6666") + ggtitle("Гістограма частот для студен
тів, які склали іспит з письма")
```




```
summary(df[df$Group==0,])
```

```
##    gender        race/ethnicity    parental level of education
## Length:656       Length:656        Length:656
## Class :character   Class :character   Class :character
## Mode  :character   Mode  :character   Mode  :character
##
##
##
##    lunch         test preparation course   math score     reading score
## Length:656       Length:656           Min.   : 15.00   Min.   : 25.0
## Class :character   Class :character       1st Qu.: 57.00   1st Qu.: 58.0
## Mode  :character   Mode  :character       Median : 67.00   Median : 68.5
##                                  Mean   : 66.49   Mean   : 68.1
##                                  3rd Qu.: 78.00   3rd Qu.: 78.0
##                                  Max.   :100.00   Max.   :100.0
## writing score       Group       X1_1           X2
## Min.   : 15.00   Min.   :0   Min.   : 15.00   Min.   : 25.0
## 1st Qu.: 56.00   1st Qu.:0   1st Qu.: 57.00   1st Qu.: 58.0
## Median : 66.00   Median :0   Median : 67.00   Median : 68.5
## Mean   : 65.64   Mean   :0   Mean   : 66.49   Mean   : 68.1
## 3rd Qu.: 76.00   3rd Qu.:0   3rd Qu.: 78.00   3rd Qu.: 78.0
## Max.   :100.00   Max.   :0   Max.   :100.00   Max.   :100.0
```

```
summary(df[df$Group==1,])
```

```
##    gender       race/ethnicity    parental level of education
##  Length:344       Length:344       Length:344
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##    lunch        test preparation course   math score     reading score
##  Length:344       Length:344          Min.   : 27.00   Min.   : 38.00
##  Class :character   Class :character      1st Qu.: 60.75   1st Qu.: 66.00
##  Mode  :character   Mode  :character      Median : 70.00   Median : 75.00
##                                Mean   : 70.33   Mean   : 74.73
##                                3rd Qu.: 81.00   3rd Qu.: 84.00
##                                Max.   :100.00   Max.   :100.00
##  writing score      Group       X1_1          X2
##  Min.   : 41.00   Min.   :1   Min.   : 27.00   Min.   : 38.00
##  1st Qu.: 67.00   1st Qu.:1   1st Qu.: 60.75   1st Qu.: 66.00
##  Median : 75.00   Median :1   Median : 70.00   Median : 75.00
##  Mean   : 75.81   Mean   :1   Mean   : 70.33   Mean   : 74.73
##  3rd Qu.: 86.00   3rd Qu.:1   3rd Qu.: 81.00   3rd Qu.: 84.00
##  Max.   :100.00   Max.   :1   Max.   :100.00   Max.   :100.00
```

```
# Дисперсія змінної
var(df[df$Group==0,10:11])
```

```
##       X1_1     X2
## X1_1 236.5494 178.0228
## X2   178.0228 199.0213
```

```
var(df[df$Group==1,10:11])
```

```
##       X1_1     X2
## X1_1 215.9025 151.9342
## X2   151.9342 170.7356
```

```
df <- df[,9:11]
```

```
set.seed(123)
```

```
#використаємо 80% набору даних як набір для навчання та 20% як набір для тестування
sample <- sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.8,0.2))
train  <- df[sample, ]
test   <- df[!sample, ]
```

```
dim(train)
```

```
## [1] 802   3
```

```
dim(test)

## [1] 198  3

############  Random forests

#set.seed(51)

#ii <- createDataPartition(df[,1:3], p=.75, list=F)
#xTrain <- df[ii, 2:3] # predictors for training
#yTrain <- df[ii, 1] # class label for training
#xTest <- df[-ii, 2:3] # predictors for testing
#yTest <- df[-ii, 1] # class label for testing
# Random Forest
# Fit Random Forest model
# Fix ntree and mtry
#str(xTrain)
#mdl_RF <- train(x=xTrain,y=as,factor(yTrain),method='rf',ntree=200,tuneGrid=dat
a.frame(mtry=2))

#rf <- randomForest(Group~., data=train, proximity=TRUE)
#print(rf)



# Prediction & Confusion Matrix - train data
#p1 <- predict(rf, train)
#confusionMatrix(as.numeric(p1), as.numeric(train$Group))

# Prediction & Confusion Matrix - test data
#p2 <- predict(rf, test)
#confusionMatrix(p2, test$Group) # Accuracy on test sample: 84.34%

# Error rate of Random Forest
#plot(rf, main = "Error rate of Random Forest")

# Random Search
#control <- trainControl(method="repeatedcv", number=10, repeats=3, search="rand
om")
#set.seed(113)
#mtry <- sqrt(ncol(df))
#rf_random <- train(Group~., data=df, method="rf", metric="Accuracy", tuneLength
=15,trControl=control)
#print(rf_random)
#plot(rf_random)

############  Support vector machines

library(e1071)
classifier <- svm(formula = Group ~ .,
                  data = train,
```

```
                 type = 'C-classification',
                 kernel = 'radial')

p1 <- predict(classifier, newdata = train)
p2 <- predict(classifier, newdata = test)

confusionMatrix(table(train$Group, p1))
```

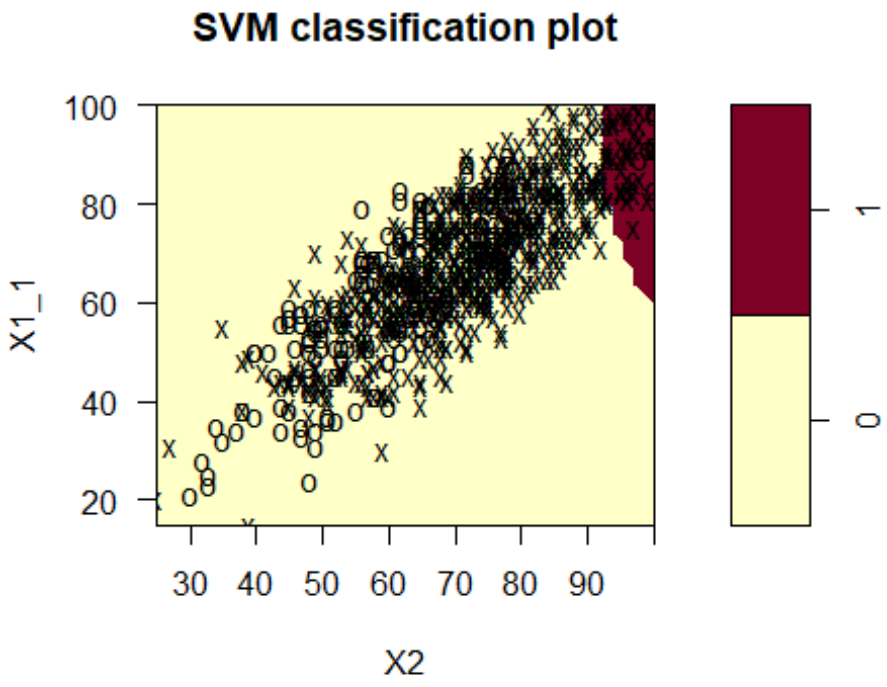
```
## Confusion Matrix and Statistics
##
##   p1
##     0   1
##  0 501  22
##  1 245  34
##
##           Accuracy : 0.6671
##             95% CI : (0.6333, 0.6997)
##    No Information Rate : 0.9302
##    P-Value [Acc > NIR] : 1
##
##              Kappa : 0.0981
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.6716
##          Specificity : 0.6071
##       Pos Pred Value : 0.9579
##       Neg Pred Value : 0.1219
##           Prevalence : 0.9302
##       Detection Rate : 0.6247
##   Detection Prevalence : 0.6521
##     Balanced Accuracy : 0.6394
##
##      'Positive' Class : 0
##
```

```
confusionMatrix(table(test$Group, p2)) # Accuracy on test sample: 86.37%
```

```
## Confusion Matrix and Statistics
##
##   p2
##     0   1
##  0 125   8
##  1  63   2
##
##           Accuracy : 0.6414
##             95% CI : (0.5704, 0.7082)
```

```
##    No Information Rate : 0.9495
##    P-Value [Acc > NIR] : 1
##
##                  Kappa : -0.0375
##
##  Mcnemar's Test P-Value : 1.468e-10
##
##             Sensitivity : 0.66489
##             Specificity : 0.20000
##          Pos Pred Value : 0.93985
##          Neg Pred Value : 0.03077
##              Prevalence : 0.94949
##          Detection Rate : 0.63131
##    Detection Prevalence : 0.67172
##       Balanced Accuracy : 0.43245
##
##       'Positive' Class : 0
##
```

```r
plot(classifier, train)
```




```r
classifier1 <- svm(formula = Group ~ .,
                   data = train,
                   type = 'C-classification',
                   kernel = 'polynomial')
```

```r
p1 <- predict(classifier1, newdata = train)
p2 <- predict(classifier1, newdata = test)

confusionMatrix(table(train$Group, p1))
```

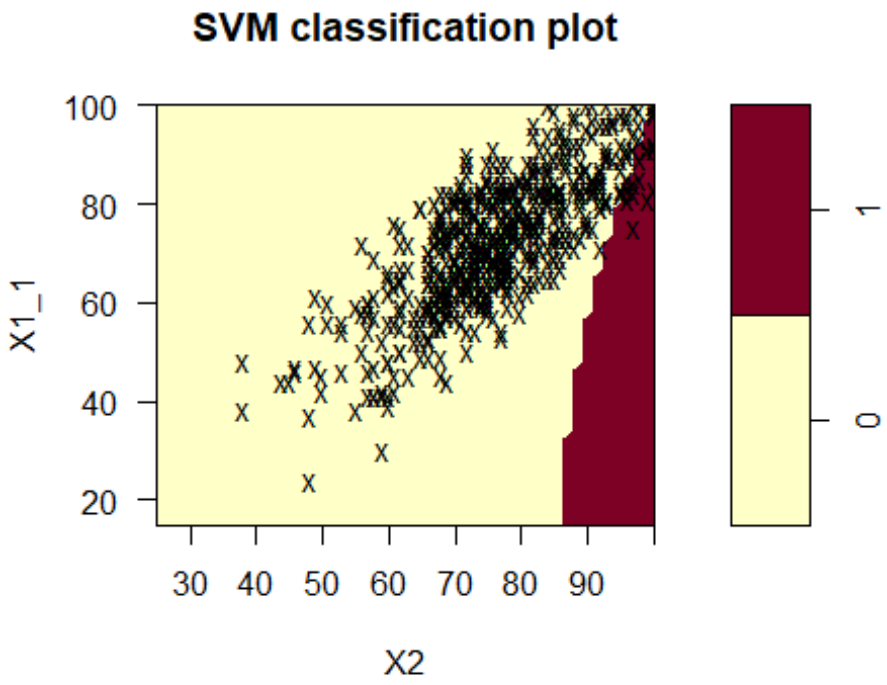
```
## Confusion Matrix and Statistics
##
##   p1
##     0   1
##  0 517   6
##  1 260  19
##
##           Accuracy : 0.6683
##             95% CI : (0.6345, 0.7009)
##    No Information Rate : 0.9688
##    P-Value [Acc > NIR] : 1
##
##              Kappa : 0.0719
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.6654
##          Specificity : 0.7600
##       Pos Pred Value : 0.9885
##       Neg Pred Value : 0.0681
##           Prevalence : 0.9688
##       Detection Rate : 0.6446
##   Detection Prevalence : 0.6521
##     Balanced Accuracy : 0.7127
##
##      'Positive' Class : 0
##
```

```r
confusionMatrix(table(test$Group, p2)) # Accuracy on test sample: 78.83%
```

```
## Confusion Matrix and Statistics
##
##   p2
##     0   1
##  0 130   3
##  1  63   2
##
##           Accuracy : 0.6667
##             95% CI : (0.5964, 0.7319)
##    No Information Rate : 0.9747
##    P-Value [Acc > NIR] : 1
##
```

```
##               Kappa : 0.0107
##
##  Mcnemar's Test P-Value : 3.803e-13
##
##             Sensitivity : 0.67358
##             Specificity : 0.40000
##          Pos Pred Value : 0.97744
##          Neg Pred Value : 0.03077
##              Prevalence : 0.97475
##          Detection Rate : 0.65657
##    Detection Prevalence : 0.67172
##       Balanced Accuracy : 0.53679
##
##       'Positive' Class : 0
##
```

```
plot(classifier1, train)
```




```
classifier2 <- svm(formula = Group ~ .,
                   data = train,
                   type = 'C-classification',
                   kernel = 'sigmoid')

p1 <- predict(classifier2, newdata = train)
p2 <- predict(classifier2, newdata = test)

confusionMatrix(table(train$Group, p1))
```

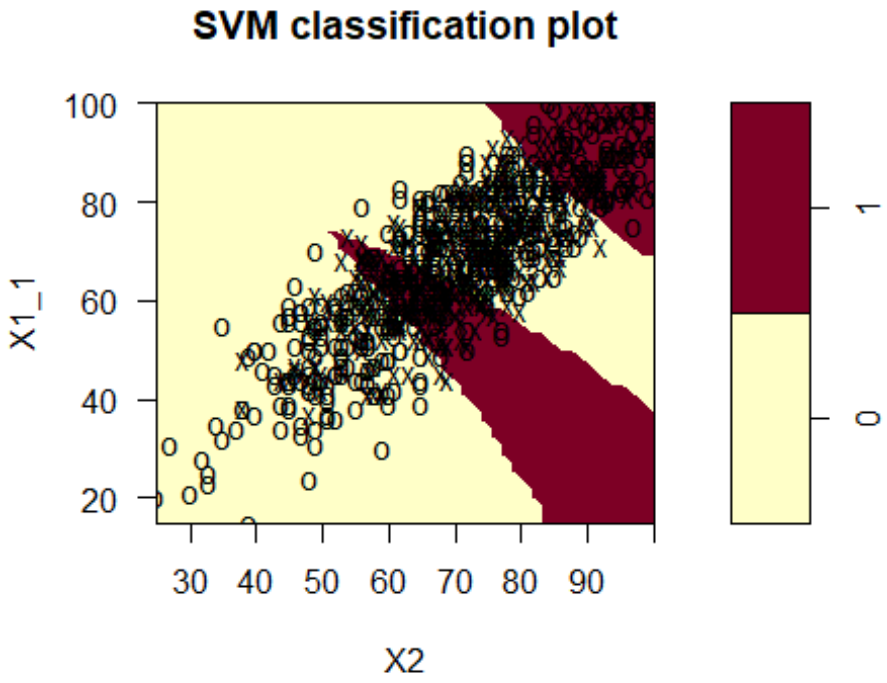
```
## Confusion Matrix and Statistics
##
##    p1
##      0   1
##   0 368 155
##   1 168 111
##
##             Accuracy : 0.5973
##               95% CI : (0.5624, 0.6314)
##    No Information Rate : 0.6683
##    P-Value [Acc > NIR] : 1.0000
##
##                Kappa : 0.1026
##
##  Mcnemar's Test P-Value : 0.5043
##
##            Sensitivity : 0.6866
##            Specificity : 0.4173
##         Pos Pred Value : 0.7036
##         Neg Pred Value : 0.3978
##             Prevalence : 0.6683
##         Detection Rate : 0.4589
##   Detection Prevalence : 0.6521
##      Balanced Accuracy : 0.5519
##
##       'Positive' Class : 0
##
```

```
confusionMatrix(table(test$Group, p2))
```

```
## Confusion Matrix and Statistics
##
##    p2
##      0  1
##   0 82 51
##   1 42 23
##
##             Accuracy : 0.5303
##               95% CI : (0.4583, 0.6014)
##    No Information Rate : 0.6263
##    P-Value [Acc > NIR] : 0.9977
##
##                Kappa : -0.0286
##
##  Mcnemar's Test P-Value : 0.4068
##
```

```
##          Sensitivity : 0.6613
##          Specificity : 0.3108
##       Pos Pred Value : 0.6165
##       Neg Pred Value : 0.3538
##           Prevalence : 0.6263
##       Detection Rate : 0.4141
##   Detection Prevalence : 0.6717
##     Balanced Accuracy : 0.4861
##
##      'Positive' Class : 0
##
```

```
plot(classifier2, train)
```



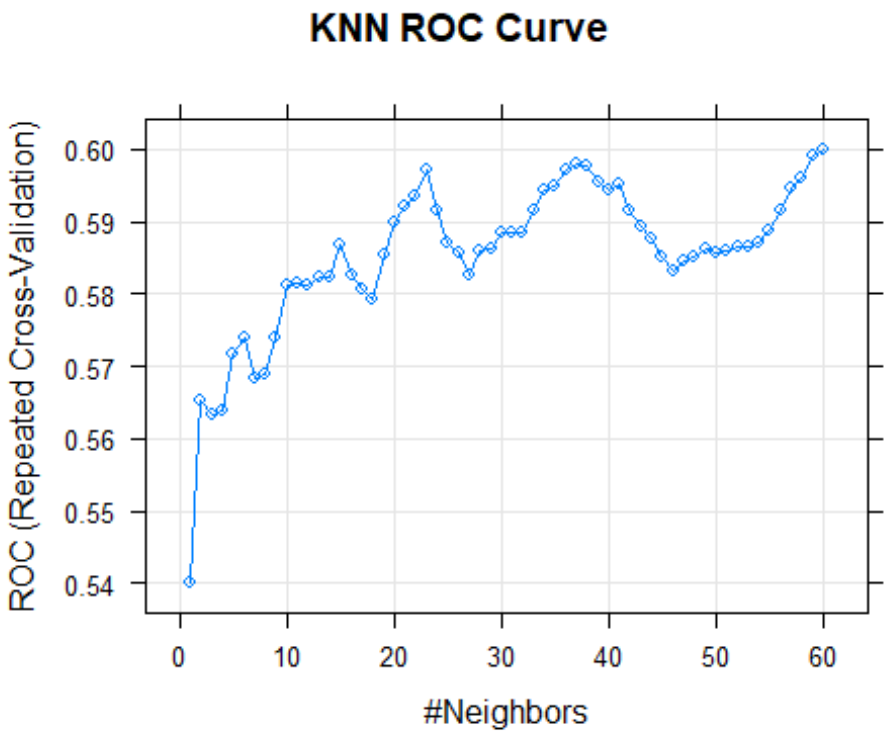

```
#############   K-nearest neighbour

dff <- df
dff$Group[dff$Group == 0] <- 'No'
dff$Group[dff$Group == 1] <- 'Yes'

set.seed(1234)
sample <- sample(c(TRUE, FALSE), nrow(dff), replace=TRUE, prob=c(0.8,0.2))
train  <- dff[sample, ]
test   <- dff[!sample, ]


trControl <- trainControl(method = "repeatedcv",
```

```
                           number = 10,
                           repeats = 3,
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)
set.seed(222)
fit <- train(Group ~ .,
             data = train,
             method = 'knn',
             tuneLength = 20,
             trControl = trControl,
             preProc = c("center", "scale"),
             metric = "ROC",
             tuneGrid = expand.grid(k = 1:60))
fit
```

```
## k-Nearest Neighbors
##
## 792 samples
##   2 predictor
##   2 classes: 'No', 'Yes'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 713, 713, 713, 712, 713, 713, ...
## Resampling results across tuning parameters:
##
##   k  ROC       Sens      Spec
##   1  0.5401065  0.6967223  0.38395062
##   2  0.5653138  0.7228713  0.36291548
##   3  0.5632157  0.7534712  0.34439696
##   4  0.5638221  0.7464441  0.31557455
##   5  0.5717782  0.7801645  0.30968661
##   6  0.5738987  0.7865627  0.26999050
##   7  0.5682898  0.7986091  0.26125356
##   8  0.5687985  0.8151911  0.24017094
##   9  0.5738777  0.8227987  0.23888889
##  10  0.5813056  0.8304306  0.21799620
##  11  0.5816316  0.8406144  0.19838557
##  12  0.5813088  0.8375181  0.18850902
##  13  0.5823154  0.8451500  0.18114910
##  14  0.5824512  0.8533986  0.16989554
##  15  0.5868204  0.8699686  0.16989554
##  16  0.5825147  0.8738268  0.15375119
##  17  0.5807852  0.8783261  0.13523267
##  18  0.5793035  0.8910982  0.13665717
##  19  0.5855030  0.8873004  0.12905983
##  20  0.5898369  0.8859942  0.11538462
```

```
##  21 0.5921824 0.9019715 0.12155745
##  22 0.5934046 0.9082608 0.12027540
##  23 0.5972584 0.9127117 0.10797721
##  24 0.5915821 0.9190856 0.10299145
##  25 0.5871397 0.9178036 0.10170940
##  26 0.5857871 0.9260885 0.09040836
##  27 0.5824960 0.9254233 0.08300095
##  28 0.5859097 0.9292453 0.08171890
##  29 0.5861266 0.9311925 0.08414055
##  30 0.5885392 0.9394896 0.08542260
##  31 0.5883740 0.9445815 0.08171890
##  32 0.5886293 0.9388728 0.07298196
##  33 0.5916522 0.9452225 0.06927825
##  34 0.5942928 0.9458999 0.06685660
##  35 0.5950435 0.9471577 0.06813865
##  36 0.5971313 0.9471577 0.06690408
##  37 0.5978966 0.9509555 0.05944919
##  38 0.5977300 0.9534470 0.06320038
##  39 0.5955443 0.9547775 0.05204179
##  40 0.5944567 0.9566884 0.05579297
##  41 0.5951438 0.9573053 0.05451092
##  42 0.5917089 0.9592163 0.05322887
##  43 0.5894313 0.9630019 0.05075973
##  44 0.5877608 0.9617320 0.05204179
##  45 0.5851024 0.9687228 0.05085470
##  46 0.5832282 0.9681060 0.05080722
##  47 0.5846064 0.9706580 0.05085470
##  48 0.5850975 0.9725689 0.05208927
##  49 0.5863040 0.9693759 0.04463438
##  50 0.5857848 0.9694001 0.04833808
##  51 0.5859191 0.9674770 0.04833808
##  52 0.5866186 0.9693880 0.05080722
##  53 0.5866584 0.9725568 0.05204179
##  54 0.5870854 0.9725931 0.05085470
##  55 0.5889078 0.9713111 0.04719848
##  56 0.5915762 0.9674891 0.04843305
##  57 0.5946154 0.9713353 0.04221273
##  58 0.5961499 0.9687833 0.04349478
##  59 0.5991978 0.9706459 0.04472934
##  60 0.5999340 0.9719279 0.03732194
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 60.
```

```r
plot(fit, main = "KNN ROC Curve")
```




```
p1 <- predict(fit, newdata = train)
p2 <- predict(fit, newdata = test)

confusionMatrix(table(train$Group, p1))
```

```
## Confusion Matrix and Statistics
##
##     p1
##       No Yes
##   No  513  10
##   Yes 255  14
##
##            Accuracy : 0.6654
##              95% CI : (0.6313, 0.6982)
##    No Information Rate : 0.9697
##    P-Value [Acc > NIR] : 1
##
##               Kappa : 0.0423
##
##  Mcnemar's Test P-Value : <2e-16
##
##          Sensitivity : 0.66797
##          Specificity : 0.58333
##       Pos Pred Value : 0.98088
##       Neg Pred Value : 0.05204
##           Prevalence : 0.96970
```

```
##        Detection Rate : 0.64773
##   Detection Prevalence : 0.66035
##      Balanced Accuracy : 0.62565
##
##       'Positive' Class : No
##
```

```
confusionMatrix(table(test$Group, p2)) # Accuracy on test sample: 84.13%
```

```
## Confusion Matrix and Statistics
##
##     p2
##      No Yes
##   No  128   5
##   Yes  68   7
##
##           Accuracy : 0.649
##             95% CI : (0.58, 0.7138)
##    No Information Rate : 0.9423
##    P-Value [Acc > NIR] : 1
##
##              Kappa : 0.0682
##
##  Mcnemar's Test P-Value : 3.971e-13
##
##          Sensitivity : 0.65306
##          Specificity : 0.58333
##       Pos Pred Value : 0.96241
##       Neg Pred Value : 0.09333
##           Prevalence : 0.94231
##       Detection Rate : 0.61538
##   Detection Prevalence : 0.63942
##      Balanced Accuracy : 0.61820
##
##       'Positive' Class : No
##
```

```
############################ Linear discriminant analysis
library(MASS)
linear <- lda(Group~., train)
linear
```

```
## Call:
## lda(Group ~ ., data = train)
##
## Prior probabilities of groups:
##      No      Yes
```

```
## 0.6603535 0.3396465
##
## Group means:
##       X1_1     X2
## No  66.06310 68.05927
## Yes 69.66543 74.32342
##
## Coefficients of linear discriminants:
##          LD1
## X1_1 -0.04932013
## X2    0.11013202
```

```
# Accuracy train data 70,33%
p1 <- predict(linear, train)$class
tab <- table(Predicted = p1, Actual = train$Group)
tab
```

```
##       Actual
## Predicted  No Yes
##     No  484 226
##     Yes  39  43
```

```
sum(diag(tab))/sum(tab)
```

```
## [1] 0.665404
```

```
# Accuracy test data 76,44%
p2 <- predict(linear, test)$class
tab1 <- table(Predicted = p2, Actual = test$Group)
tab1
```

```
##       Actual
## Predicted  No Yes
##     No  126  56
##     Yes   7  19
```

```
sum(diag(tab1))/sum(tab1)
```

```
## [1] 0.6971154
```

```
############################  Quadratic discriminant analysis

model <- qda(Group~., train)
model
```

```
## Call:
## qda(Group ~ ., data = train)
##
## Prior probabilities of groups:
##      No     Yes
```

```
## 0.6603535 0.3396465
##
## Group means:
##      X1_1     X2
## No  66.06310 68.05927
## Yes 69.66543 74.32342
```

```
# Accuracy train data 72,98%
p1 <- predict(model, train)$class
tab <- table(Predicted = p1, Actual = train$Group)
tab
```

```
##      Actual
## Predicted  No Yes
##     No  498 239
##     Yes  25  30
```

```
sum(diag(tab))/sum(tab)
```

```
## [1] 0.6666667
```

```
# Accuracy test data 76,92%
p2 <- predict(model, test)$class
tab1 <- table(Predicted = p2, Actual = test$Group)
tab1
```

```
##      Actual
## Predicted  No Yes
##     No  130  62
##     Yes   3  13
```

```
sum(diag(tab1))/sum(tab1)
```

```
## [1] 0.6875
```