Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт
з дисципліни
"Інтелектуальний аналіз даних"
для студентів
спеціальності 122 «Комп'ютерні науки»

2017

Методичні вказівки до виконання практичних робіт з дисципліни "Інтелектуальний аналіз даних" для студентів спеціальності 122 «Комп'ютерні науки» / Укл. Т.В. Федорончак. – Запоріжжя: ЗНТУ, 2017. – 44 с.

Автори: Т.В. Федорончак, к.т.н., доцент

Рецензент: В.І. Дубровін, к.т.н., професор

Відповідальний
за випуск: С.О. Субботін, к.т.н., професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол №3
від "21" листопада 2017 р.

ЗМІСТ

Вступ.....4
Практична робота №1 Знайомство з мовою R та середовищем RStudio 5
Практична робота №2 Підготовка даних та розвідувальний аналіз даних за допомогою мови R 8
Практична робота №3 Вирішення задачі класифікації за допомогою мови R.....24
Практична робота №4 Кластерний аналіз даних за допомогою мови R 35
Література.....43

ВСТУП

Метою дисципліни «Інтелектуальний аналіз даних» є вивчення методів сучасної обробки даних, аналітичного дослідження великих масивів інформації з метою виявлення нових раніше невідомих, практично корисних знань і закономірностей, необхідних для прийняття рішень.

Практичні роботи з дисципліни «Інтелектуальний аналіз даних» направлені на вивчення можливостей сучасної мови R для роботи з даними, а також вирішення задач інтелектуального аналізу даних з її допомогою.

R – це мова і програмне середовище для статистичної обробки, аналізу даних та подання даних в графічному вигляді. R є кросплатформним вільнорозповсюджуваним програмним забезпеченням та має відкриті вихідні коди. Працювати з R можна в консольному інтерфейсі або використовуючи інтегроване середовище розробки, таке як наприклад RStudio.

Базова комплектація R включає в себе функції, необхідні для статистичного аналізу та графічного відображення результатів дослідження. Для розширення функціональності використовуються пакети з бібліотеками додаткових функцій, що розповсюджуються через CRAN (Comprehensive R Archive Network).

Практична робота №1
Знайомство з мовою R та середовищем RStudio

Мета роботи

Ознайомитися з синтаксисом, структурами даних, функціями та бібліотеками мови R. Навчитися використовувати середовище RStudio для розробки мовою R.

Теоретичні відомості

R – це мова і програмне середовище для статистичної обробки, аналізу даних та подання даних в графічному вигляді. R є кросплатформним вільнорозповсюджуваним програмним забезпеченням та має відкриті вихідні коди. Це середовище складається з базового інтерпретатора мови R та окремих пакетів, які реалізують спеціальні методи та технології обробки даних. Завдяки наявності великої кількості безкоштовних бібліотек розширень під різноманітні задачі, останніми роками ця мова здобуває все більшу популярність серед тих хто займається аналізом даних. Мова R є безкоштовним конкурентом Matlab. Також зіставляє конкуренцію в даній галузі мові програмування Python.

Працювати з R можна в консольному інтерфейсі або використовуючи інтегроване середовище розробки, таке як наприклад RStudio.

RStudio – це інтегроване середовище розробки для мови R. Воно містить консоль для виконання команд, надає інтерфейс для доступу к історії введених команд, містить засоби побудови графіків, редактор сценаріїв, підтримує відлагодження коду та управління робочим простором, в якому зберігаються дані та програми, завантажені під час сесії роботи з середовищем. RStudio має версію, що вільно розповсюджується для різних операційних платформ.

На рис. 1.1 наведено графічний інтерфейс середовища RStudio.

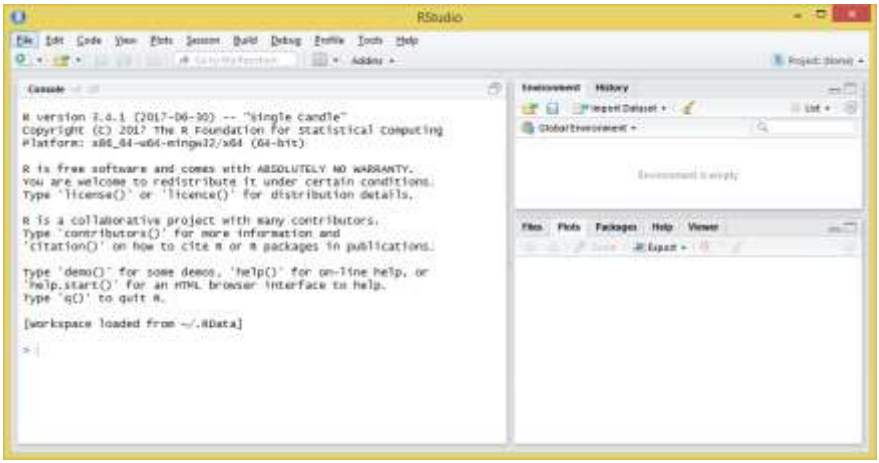


Рисунок 1.1 – Середовище RStudio

Завдання

1. Встановити програмне середовище R (https://www.r-project.org/). На жовтень 2017 року остання версія R 3.4.2.
2. Встановити середовище розробки RStudio (https://www.rstudio.com/)
3. В середовищі RStudio за необхідності змінити робочу директорію:
   - в консолі за допомогою команди setwd();
   - в меню Session -> Change Working Directory.
4. Встановити бібліотеку swirl (http://swirlstats.com/):
   - в консолі за допомогою команди install.packages("swirl");
   - в меню Tools->Install Packages.

Бібліотека swirl – це стрімке занурення в R (learning by doing) або «Learn R, in R».

5. Якщо в Windows не виходить встановити бібліотеки, тому що немає доступу на запис в папці Program Files, в робочій директорії створити файл з назвою .Renviron і в ньому вказати свою папку для бібліотек командою R_LIBS_USER = D:/myRlib.
6. Завантажити бібліотеку в середовище, встановити курси та запустити бібліотеку:

```
> library(swirl)
```

```
> install_course("R Programming E")
> install_course("Data Analysis")
> swirl()
```

7. Пройти курси, що було завантажено для знайомства з мовою програмування R.

8. Для подальшого знайомства з мовою можна скористатися книгою «Мастицкий С. Э., Шитиков В. К. Статистический анализ и визуализация данных с помощью R», С. 23-96.

Контрольні питання

1. Дайте коротку характеристику мови програмування R.
2. Які базові типи даних існують в мові R.
3. Опишіть базовий синтаксис мови R (створення змінних, привласнення, керуючі конструкції, функції).
4. Що таке робочий простір та робоча директорія?
5. Як створити та запустити сценарій?
6. Як завантажити дані з файлів?
7. Опишіть особливості типу даних data frame.
8. Виконання обчислень над векторами за допомогою apply-функцій.
9. Базові графічні можливості мови R.

Практична робота №2
Підготовка даних та розвідувальний аналіз даних за допомогою мови R

Мета роботи

Навчитися використовувати мову R для підготовки даних до аналізу та виконання розвідувального аналізу даних.

Теоретичні відомості

Розвідувальний аналіз даних (Exploratory data analysis) – це попередній аналіз, який виконується з метою виявлення основних властивостей даних, знаходження в них найбільш загальних закономірностей, законів розподілу та тенденцій, виявлення аномалій, побудови початкових моделей. Розвідувальний аналіз використовує статистичні методи та методи графічного аналізу.

Залежності, які відшукуються в даних, визначаються поставленими питаннями дослідження, тобто гіпотезами, що висуваються.

Зазвичай в результаті розвідувального аналізу вирішується чи достатньо даних, які є в наявності, для проведення подальших досліджень, чи необхідно зібрати якісь додаткові дані для вирішення поставленого завдання, чи необхідно уточнити постановку завдання дослідження.

R надходить з декількома вбудованими наборами даних, які зазвичай використовуються для демонстрації можливостей мови та прикладів вирішення стандартних задач аналізу даних.

Для того, щоб побачити перелік з близько 90 існуючих наборів даних, необхідно скористатися функцією data(). Для завантаження одного з наборів в середовище необхідно викликати функцію data(dataset_name), указавши в якості аргументу назву набору даних.

Щоб отримати опис обраного набору даних необхідно викликати функцію help(dataset_name).

Найбільш часто в наукових дослідженнях для перевірки якості розроблених нових методів аналізу використовуються такі набори даних як:

– mtcars – дані з журналу «US Motor Trend» за 1974 рік, що

включають витрати палива та 10 аспектів дизайну для 32 автомобілів;

– iris – набір даних з вимірюваннями розмірів квіток ірису трьох сортів;

– ToothGrowth –результати дослідження впливу вітаміну С на ріст зубів морських свинок;

– PlantGrowth – результати експерименту для порівняння врожайності (вимірюється сухою вагою рослин), що отримані під контролем та двома різними умовами обробки;

– USArrests – статистичні дані про насильницькі злочини в США.

Наприклад, після виклику функції data(iris) в робочому просторі середовища R з'явиться змінна iris, що містить набор даних про розміри квіток ірису.

Іриси Фішера – це багатовимірний набір даних для задачі класифікації, на прикладі якого англійський статистик та біолог Рональд Фішер в 1936 році продемонстрував роботу розробленого ним методу дискримінантного аналізу.

Дані складаються з 150 вимірювань ірисів з трьох видів — Iris setosa, Iris virginica і Iris versicolor, по 50 вимірювань на вид. Для кожного екземпляра вимірювалися чотири характеристики в сантиметрах: довжина та ширина чашолистка (Sepal length and width) та довжина та ширина пелюстки (petal length and width).

За даними вимірів будують правила класифікації, що дозволяє визначити вид рослини за даними вимірювань. Це задача багатокласової класифікації. Один з класів (Iris setosa) набору даних є лінійно-виокремленим від двох інших.

Рисунок 2.1 – Іриси Фішера

Дані для аналізу зазвичай попередньо готуються в табличних форматах у вигляді txt або csv файлів.

Для завантаження файлу з даними необхідно розмістити його в робочому каталозі програми. Для визначення поточного робочого каталогу використовується функція getwd(), а для встановлення нового каталогу – функція setwd(path).

Щоб отримати допомогу щодо призначення та параметрів виклику будь-якої функції мови R необхідно викликати команду довідки в одному з наведених форматів:

```
> help(function)
> ?function
```

Для завантаження власних наборів даних в середовище R використовується функції read.table() та read.csv(). Наприклад,

```
> my_data <- read.table(file = "mydata.txt", sep=";", head=TRUE)
> my_data <- read.csv(file = "mydata.csv", header = TRUE)
```

Щоб попередньо переглянути файл для завантаження використовується команда file.show("mydata.txt").

Для зберігання наборів даних в R використовуються табличний тип даних (data frame). Кожен стовпець таблиці є вектором, що містить дані певного типу. Він представляє атрибут деякого об’єкту. Рядок таблиці – це об’єкт вибірки даних. Всі стовпці повинні мати однакову довжину. Кожен стовпець та кожний рядок може мати ім’я. Для того, щоб отримати доступ до компонентів таблиць можна використовувати знак $, квадратні дужки з вказівкою двох індексів [номер_рядка, номер_стовпця], подвійні квадратні дужки [[]] або безпосередньо ім’я стовпця.

Нехай є таблиця my_data, що описує кількість населення міст та містить три стовпці City, Sex та Number. Звернутися до певної частини таблиці можна наступним чином:

```
> #стовпець з ім’ям Number
> my_data$Number
> #другий стовпець таблиці, тобто Sex
> my_data[,2]
> #третій стовпець, тобто Number
```

```
> my_data[[3]]
> #стовпець з ім’ям Number
> my_data["Number"]
> # 4-й елемент зі стовпця Number
> my_data$Number[4]
> # елементи 1-3 зі стовпця Number
> my_data$Number[1:3]
> # елементи 1, 3, 4 зі стовпця Number
> my_data$Number[c(1, 3, 4)]
> # всі значення численності, що перевищують 10000
> my_data$Number[my_data$Number > 10000]
> # всі значення численності чоловічого населення
> my_data$Number[my_data$Sex == "Male"]
```

Після завантаження даних необхідно перевірити розмір та структуру даних. Розмірність даних та імена даних можуть бути отримані за допомогою функцій dim() та names(). Функцій str() та attributes() повертають структуру та атрибути даних.

```
> str(iris)
'data.frame':         150 obs. of  5 variables:
$ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
$ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 ...
```

Для того, щоб визначити типи атрибутів, можна скористатися наступним кодом:

```
> sapply(iris, class)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width     Species
   "numeric"   "numeric"   "numeric"   "numeric"    "factor"
```

Змінна, що визначає клас об’єктів, повинна мати тип factor, тобто її значення повинні мати декілька рівнів або міток. Щоб побачити значення, що приймає атрибут класу, використаємо функцію levels():

```
> levels(iris$Species)
[1] "setosa"     "versicolor" "virginica"
```

Далі необхідно ознайомитися зі структурою рядків даних. Щоб отримати перші або останні рядки даних використовуються функції head() або tail(). Також можна отримати перші три рядки даних наступним чином:

```
> iris[1:3,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2            3         0.2  setosa
```

Можна отримати значення однієї колонки даних:

```
> iris[1:10, "Sepal.Length"]
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

або

```
> iris$Sepal.Length[1:10]
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

Щоб перевірити чи має змінна пропущені значення (NA), використовується функція is.na(x). Можна підрахувати кількість пропущених значень у векторі sum(is.na(x)), або отримати всі заповнені значення x[!is.na(x)].

Щоб отримати лише ті об'єкти таблиці, що не мають пропущених значень використовується функція complete.cases(). Наприклад:

```
> good <- complete.cases(iris)
```

Щоб отримати розподіл кожної чисельної величини використовується функція summary(). Ця функція повертає мінімум, максимум, середнє, медіану, перший та третій квартилі. Для категоріальних змінних дана функція повертає частоту кожного значення.

```
> summary(iris$Sepal.Length)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800    5.843   6.400   7.900
```

Для отримання середнього, медіани та діапазонів використовуються функції mean(), median() та range(). Отримати квантилі можна за допомогою функції quantile(). Функція var() повертає дисперсію.

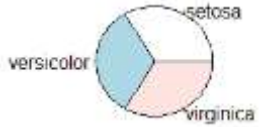
Перевірити розподіл змінної можна відобразивши гістограму та щільність розподілу за допомогою функцій hist() та plot(density()).

Для отримання характеристики категоріальної змінної можна скористатися функцією table(). Наприклад:

```
> table(iris$Species)
    setosa versicolor  virginica
        50         50         50
```

Після чого можна відобразити секторну діаграму або стовпчасту функціями pie() або barplot().

```
> pie(table(iris$Species))
```



Щоб отримати всі унікальні значення, які приймає змінна, необхідно скористатися функцією unique(). Наприклад:

```
> unique(iris$Sepal.Width)
[1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 2.9 3.7 4.0 4.4 3.8 3.3 4.1
[15] 4.2 2.3 2.8 2.4 2.7 2.0 2.2 2.5 2.6
```

Дослідивши розподіл окремих величин, виконується дослідження зв'язків між парами змінних. Функції cov() та cor() розраховують коваріацію та кореляцію змінних. Наприклад:

```
> cor(iris[,1:4])
              Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Базовим підходом при проведенні наукового дослідження є

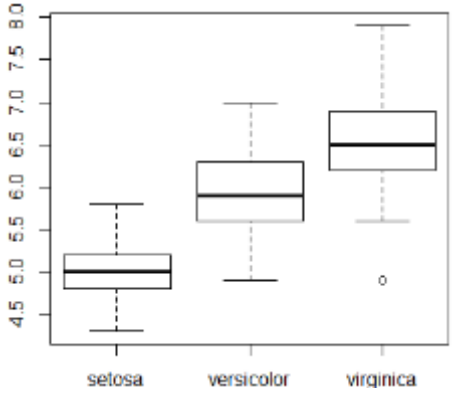
порівняння або **протипоставлення**. Наприклад, при підтвердженні гіпотези завжди існує альтернативна гіпотеза, що приймається у разі відкидання основної гіпотези. Так само при визначенні характеристик даних визначення загальних характеристик по всій вибірці може бути непоказовим. Більш цікавим є визначення характеристик даних в залежності від значення деякого іншого атрибуту, що може дати можливість провести порівняння. Мета порівняння залежить від гіпотез, що висуває дослідник щодо даних.

Для розрахунку статистики певної змінної в залежності від значення деякої категоріальної змінної (наприклад, класу в даних) використовується функція aggregate():

```
> aggregate(Sepal.Length ~ Species, summary, data=iris)
     Species     Min.   1st Qu.   Median     Mean   3rdQu.    Max.
1     setosa    4.300    4.800    5.000    5.006    5.200    5.800
2 versicolor    4.900    5.600    5.900    5.936    6.300    7.000
3  virginica    4.900    6.225    6.500    6.588    6.900    7.900
```

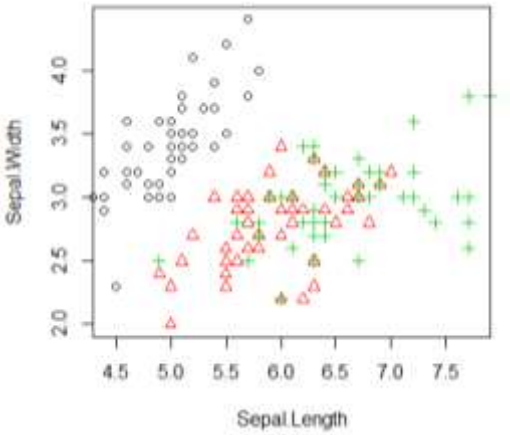
Для побудови статистичного графіку, що має назву ящик з вусами, використовується функція boxplot(). Наприклад:

```
> boxplot(Sepal.Length~Species, data=iris)
```



Для відображення діаграм розкиду використовується функція plot().

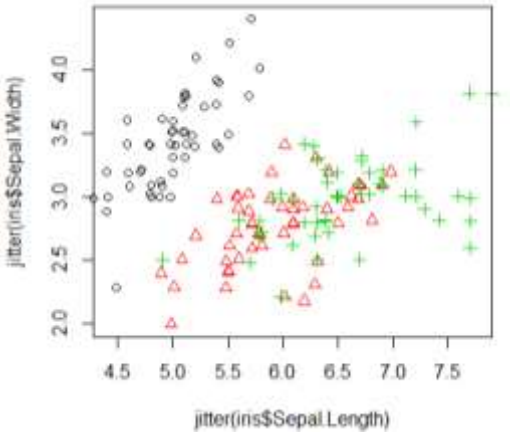
```
> with(iris, plot(Sepal.Length, Sepal.Width, col=Species,
pch=as.numeric(Species)))
```



В наведеному прикладі функція with() дозволяє вказати набір даних, з яким виконується робота.

Інколи при побудові діаграми розкиду в даних можу бути декілька об’єктів з однаковими характеристиками. Такі об’єкти на діаграмі будуть перекривати один одного. Щоб додати незначний шум і побачити це перекриття використовується функція jitter(). Наприклад,

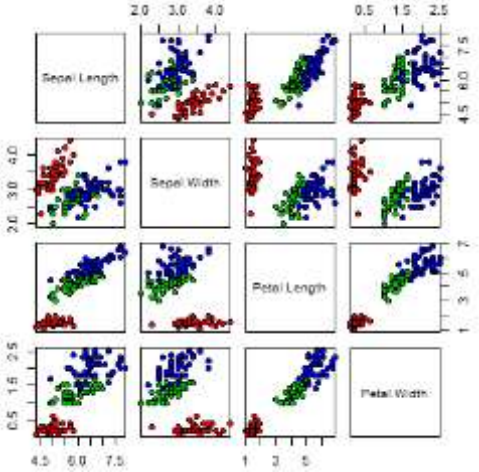
```
> plot( jitter( iris$Sepal.Length ), jitter( iris$Sepal.Width ))
```



Матриця з усіма можливими комбінаціями пар змінних на

діаграмах розкиду може бути побудована за допомогою функції pairs(). Наприклад,

```
> pairs(iris[1:4], pch = 21, bg = c("red", "green3",
"blue")[unclass(iris$Species)])
```



Зверніть увагу на те, що всі діаграми, що будуються для даних, повинні мати підписи та анотації, що пояснюють їх зміст.

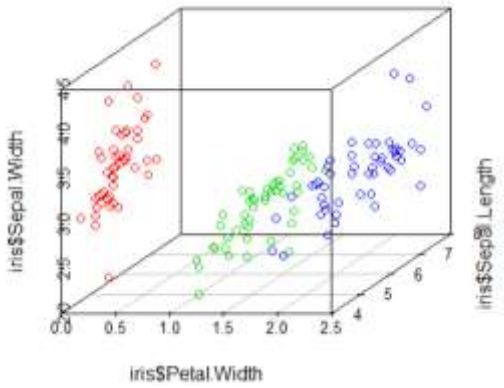
Завантаживши додаткові бібліотеки, що розширюють базові графічні функції R, можна отримати більш складне візуальне відображення даних. Щоб встановити додаткові бібліотеки, необхідно виконати наступну команду

```
> install.packages("package name")
```

або в інтерфейсі RStudio обрати пункт меню Tools>Install Package.

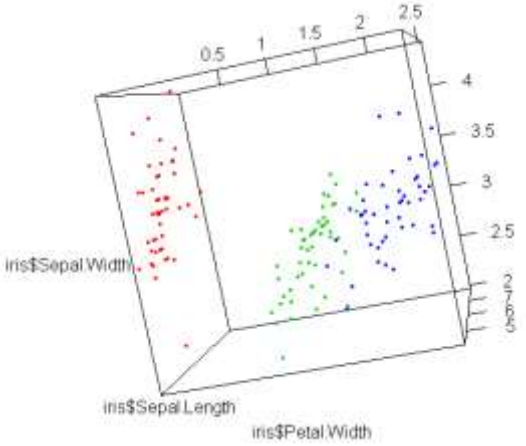
Наприклад, можна відобразити тривимірне подання даних:

```
> library(scatterplot3d)
> scatterplot3d(iris$Petal.Width, iris$Sepal.Length, iris$Sepal.Width,
color = c("red", "green3", "blue")[unclass(iris$Species)])
```



або

```
> library(rgl)
> plot3d(iris$Petal.Width, iris$Sepal.Length, iris$Sepal.Width,
col=c("red", "green3", "blue")[unclass(iris$Species)])
```



Для дослідження даних широко застосовується бібліотека **ggplot2**. Це найбільш потужна бібліотека для створення найрізноманітніших графіків.

ggplot2 заснований на «графічній граматиці», тобто на ідеї, згідно з якою будь-який графік можна побудувати з одних і тих же складових: масиву даних (data), системи координат і геометричних

об'єктів (geom) – візуальних маркерів, що відображають окремі значення спостережень (точки, лінії, стовпці і т.д.). Значення наносяться на графік за допомогою візуальних властивостей геометричних об'єктів (Aesthetics), таких як розмір, колір, координати x і y.

Найбільш потужною функцією цього пакету є ggplot(). Опис цієї функції можна знайти за посиланням https://r-datascience.ru/ggplot2_guide/

Функція qplot() використовується для швидкого налаштування параметрів графіка. ЇЇ формат:

qplot(x, y, data=, facets=, geom=, xlim=, ylim= xlab=, ylab=, main=)

де data – набір даних для відображення;

x, y – змінні, що будуть відображені по горизонтальній та вертикальній вісі, для одномірних графіків, таких як гістограми, параметр y пропускається;

facets – панелі для відображення декількох графіків, кожний з яких побудований для підмножени даних; може бути заданий у вигляді формули: rowvar ~ colvar, rowvar~. або .~colvar.

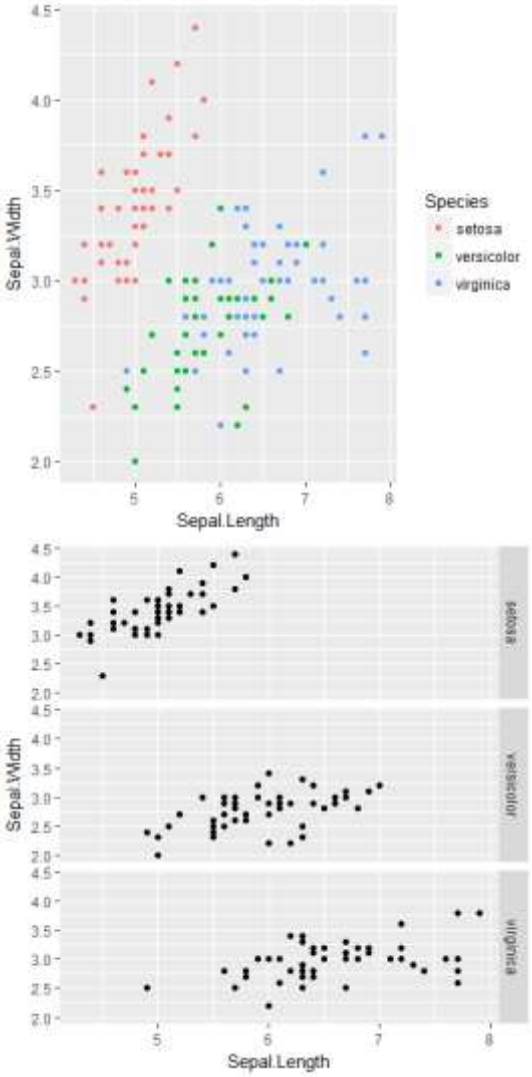
geom – геометрія об’єктів, що визначає тип графіку; цей параметр приймає вектор строкових параметрів: "point", "smooth", "boxplot", "line", "histogram", "density", "bar", "jitter";

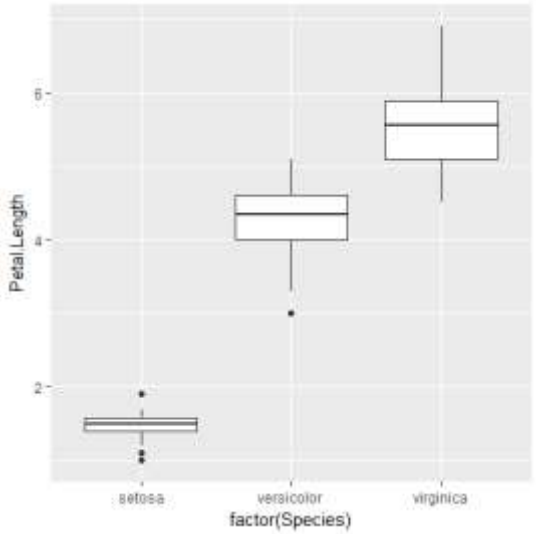
xlim,ylim – межі графіків;

main, xlab, ylab – строкові вектори підписів.

Наприклад,

```
>#використання бібліотеки ggplot2 для побудови графіків
> ggplot(iris, aes(Sepal.Length, Sepal.Width, colour = Species)) +
geom_point()
> qplot(Sepal.Length, Sepal.Width, data=iris, facets=Species ~.)
> qplot(factor(Species), Petal.Length, data = iris, geom = c("boxplot"))
```


Sepal.Width
Sepal.Length
Species
setosa
versicolor
virginica
4.5
4.0
3.5
3.0
2.5
2.0
5
6
7
8



Для роботи з даними у вигляді таблиць широко використовується бібліотеку **dplyr**. Він надає наступні функції для роботи з даними:

– select – підвибірка стовпців даних;

– filter – підвибірка рядків даних з використанням логічних умовних виразів;

– arrange – зміна порядку рядків в даних;

– mutate – додавання нових стовпців шляхом перетворення існуючих;

– summarise – групування даних та підведення підсумків.

Наприклад,

```
> library(dplyr)
> #обрати лише ознаки ip_id, package, country
> select(cran, ip_id, package, country)
> #обрати ознаки послідовно від r_arch до country
> select(cran, r_arch:country)
> #обрати всі ознаки, крім ознак від X до size
> select(cran, -(X:size))
> #вибрати об’єкти, в яких ознака package = "swirl"
> filter(cran, package == "swirl")
> #вибрати об’єкти, в яких r_version = "3.1.1" та country = "US"
```

```
> filter(cran, r_version == "3.1.1", country == "US")
> #вибрати об’єкти, в яких country ="US" або  country = "IN"
> filter(cran, country =="US" | country == "IN")
> #вибрати об’єкти, в яких , size>100500 та  r_os = "linux-gnu"
> filter(cran, size>100500, r_os == "linux-gnu")
> #вибрати об’єкти, в яких значення ознаки r_version заповнено
> filter(cran, !is.na(r_version))
> #відсортувати об’єкти за спаданням значення ознаки ip_id
> arrange(cran2, desc(ip_id))
> #відсортувати об’єкти за зростанням значенням ознаки package,
а потім за зростанням ip_id
> arrange(cran2, package, ip_id)
> arrange(cran2, country, desc(r_version), ip_id)
> #створити нові ознаки
> mutate(cran3, size_mb = size / 2^20, size_gb = size_mb/2^10)
> mutate(cran3, correct_size = size + 1000)
> #підбити підсумки в даних
> summarize(cran, avg_bytes = mean(size))
```

Функція summarize зазвичай використовується на даних, що були згруповані за допомогою функції group_by. Тоді результат роботи функції буде відображено для кожної групи.

Функція group_by використовується для групування даних:

```
> by_package<-group_by(cran, package)
> summarise(by_package, mean(size))
> pack_sum <- summarize(by_package,
                count = n(),
                unique = n_distinct(ip_id),
                countries = n_distinct(country),
                avg_bytes = mean(size))
```

Функції бібліотеки dplyr можуть вкладатися одна в одну, наприклад:

```
> result2 <-
 arrange(
   filter(
    summarize(
      group_by(cran,
            package
```

```
    ),
    count = n(),
    unique = n_distinct(ip_id),
    countries = n_distinct(country),
    avg_bytes = mean(size)
   ),
   countries > 60
  ),
  desc(countries),
  avg_bytes
 )
```

Завдання

1. Обрати один із вбудованих в середовище мови R набір даних.
2. Описати практичну задачу, що описує обраний набір. Описати атрибути та кількість об’єктів в вибірці. Чи є в даних цільовий атрибут?
3. Провести розвідувальний аналіз даних та дати характеристику даних.

Контрольні питання

1. В чому полягає початкова попередня обробка даних для аналізу?
2. Що таке розвідувальний аналіз даних.
3. Що таке генеральна сукупність та вибірка?
4. Що таке репрезентативна вибірка?
5. Що таке гіпотеза?
6. Яким чином можна завантажити дані в мові R?
7. Опишіть особливості формату даних data frame. Як з ним працювати?
8. Як перевірити розмір та структуру даних в мові R?
9. Як вивести деякі об’єкти з вибірки в мові R?
10. Як визначити, чи містить вибірка пропущені значення? Як підрахувати кількість пропущених значень змінної?
11. Як отримати підсумкові характеристики кожного атрибуту даних?

12. Що таке мінімум, максимум, математичне очікування, мода та медіана випадкової величини? Як їх отримати за допомогою мови R?

13. Що таке квантиль та квартиль випадкової величини? Як їх отримати за допомогою мови R?

14. Що відображає гістограма та графік щільності розподілу величини? Як їх побудувати за допомогою мови R?

15. Що таке коваріація та кореляція двох величин? Як їх отримати за допомогою мови R?

16. Що відображає графік, що має назву ящик з вусами? Як його побудувати за допомогою мови R?

17. Як побудувати діаграми розкиду даних?

18. Опишіть можливості бібліотеки ggplot2. Для чого вона використовується та які задачі дозволяє вирішити?

19. Опишіть можливості бібліотеки dplyr. Для чого вона використовується та які задачі дозволяє вирішити?

20. Як виконати підвибірку стовпців даних за допомогою функції select ?

21. Як виконати підвибірку рядків даних з використанням логічних умовних виразів за допомогою функції filter?

22. Як змінити порядок рядків в даних за допомогою функції arrange?

23. Як додати нові стовпці в дані за допомогою функції mutate?

24. Як згрупувати дані та отримати підсумкові значення за допомогою функції summarise? Для чого використовується функція group_by?

Практична робота №3
Вирішення задачі класифікації за допомогою мови R

Мета роботи

Навчитися використовувати мову R для вирішення задачі класифікації.

Теоретичні відомості

Задача класифікації полягає в побудові моделі, яка дозволяє прогнозувати приналежність об'єкту до одного із визначених класів на основі значень атрибутів, що характеризують цей об'єкт. Вона відноситься до навчання з учителем.

Для того, щоб вирішити задачу класифікації даних, необхідно підготувати дві вибірки із заздалегідь класифікованими об'єктами. Одна з вибірок – *навчальна вибірка* – використовується для побудови класифікаційної моделі. Інша вибірка – *тестова вибірка* – використовується для оцінки якості побудованої моделі.

Існує два підходи для розбиття початкової вибірки на дві частини.

Один із них полягає у випадковому розбитті вибірки на дві частини у деякому відсотковому відношенні (80/20, 70/30, 60/40 тощо). Наприклад, можна розбити дані таким чином, щоб 2/3 частини об'єктів опинилися в навчальній вибірці, а решта 1/3 об'єктів – в навчальній.

Інший підхід має назву перехресної перевірки (cross-validation). Дані, що є в наявності, розбиваються на k блоків однакового розміру. Далі k-1 частина використовується для навчання моделі, а решта – для тестування. Процедура повторюється k разів так, щоб кожний раз для тестування використовувався новий блок даних, а отримані результати усереднюються. Leave-one-out – це спеціальний випадок перехресної перевірки, при якому кількість блоків дорівнює кількості примірників у вибірці. Таким чином, кожен раз тільки один примірник виступає в якості тестової вибірки.

Зазвичай, якщо початкова вибірка є великою, використовується відсоткове розбиття даних, якщо ж вибірка є малою, використовується перехресна перевірка.

Навчальна та тестові вибірки повинні бути стратифіковані, тобто в них повинні зберегтись початкові пропорції класів об’єктів.

Розглянемо приклад розбиття вибірки на дві частини в пропорції 60/40.

Для надання можливості повторення експериментів, пов’язаних з генерацією випадкових чисел (наприклад, побудова випадкових вибірок), в мові R необхідно спочатку задати початкове число (seed) для генератора випадкових чисел. В результаті при використанні однакового початкового числа генератор кожного разу буде генерувати ту саму послідовність випадкових чисел.

```
> set.seed(1234)
```

Після цього за допомогою функції sample() можна сформувати підвибірку

```
> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.6, 0.4))
> # створимо навчальну вибірку
> iris.training <- iris[ind==1, 1:4]
> # мітки класу для навчальної вибірки
> iris.trainLabels <- iris[ind==1,5]
> # створимо тестову вибірку
> iris.test <- iris[ind==2, 1:4]
> # мітки класу для тестової вибірки
> iris.testLabels <- iris[ind==2, 5]
```

Середовище статистичних обчислень R відрізняється особливо високою кількістю реалізованих в ній алгоритмів машинного навчання. Однак разом з перевагою наявності такої великої кількості алгоритмів виникають і деякі проблеми. По-перше, для аналітика стає все складніше пам'ятати особливості застосування та синтаксису відповідних R-функцій. По-друге, функції, що реалізують різні моделі, часто розкидані по різних бібліотеках. У зв'язку з цим було виконано спробу розробити універсальний інтерфейс, що надає доступ до основних алгоритмів машинного навчання, реалізованих в R і інших спеціалізованих статистичних системах (наприклад, Weka). Результатом цієї роботи став пакет **CARET** (Classification And Regression Training).

Для вирішення задачі класифікації мовою R скористаємося

бібліотекою caret. Ця бібліотека реалізує велику кількість функцій необхідних для вирішення задач навчання з учителем. Вона надає засоби для розбиття даних, візуалізації, попередньої обробки даних, відбору ознак, визначення значимості ознак, побудови та оцінки класифікаційних та регресійних моделей. Офіційна сторінка бібліотеки https://topepo.github.io/caret/.

Особливостями пакету caret є:

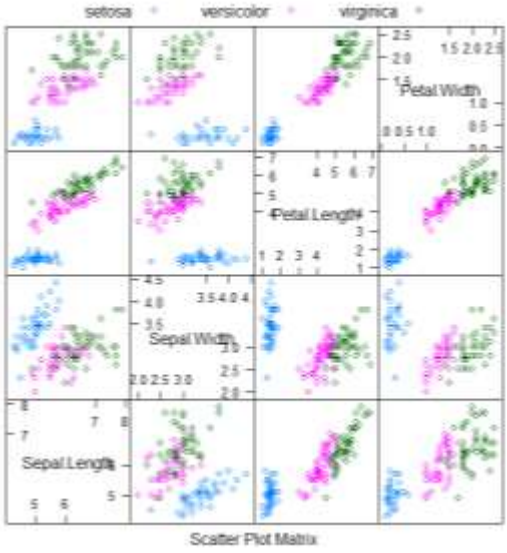
– використання універсального синтаксису команд, незалежно від синтаксису вихідної функції, що реалізує той чи інший алгоритм;

– автоматизоване знаходження оптимальних значень параметрів моделей (tuning parameters), які зазвичай неможливо обчислити аналітично;

– можливість організації паралельних обчислень, що значно прискорює процес навчання моделей.

Наприклад, для візуального подання даних в бібліотеці caret використовується функція featurePlot().

```
> library(caret)
> featurePlot(x = iris[, 1:4],  y = iris$Species,  plot = "pairs", auto.key =
list(columns = 3))
```



Розбиття даних на навчальну та тестову підвибірки можна виконати за допомогою функції createDataPartition(). Наприклад, якщо необхідно розбити дані в пропорції 75% та 25%:

```
> set.seed(107)
> trainIndx <- createDataPartition(y = iris$Species, p=0.75, list=FALSE)
> trainSet <- iris[trainIndx,]
> testSet <- iris[-trainIndx,]
```

За замовчанням createDataPartition виконує стратифіковану випадкову вибірку даних.

Для попередньої обробки та підготовки даних для аналізу бібліотека має декілька функцій. Розглянемо найбільш цікаві з них.

Функція nearZeroVar() дозволяє виявити атрибути, дисперсія значень яких близька до нуля. Як правило, такі змінні слід виключити з подальшого розгляду, тому що вони несуть дуже мало корисної інформації.

Функція findCorrelation() дозволяє виявити атрибути, які в значній мірі корелюють з іншими атрибути. Деякі моделі (наприклад, статистичні) дуже чутливі до наявності подібних змінних і дають нестійкі рішення. Тому атрибути, які високо корелюють один з одним, слід виключити з даних.

Функція preProcess() виконує задані користувачем перетворення вихідних значень атрибутів, необхідні для коректного використання тих чи інших алгоритмів навчання моделей. Зокрема, є можливість виконати стандартизацію значень числових змінних, перетворення вихідних даних методом головних компонент (що є альтернативою згаданому вище видаленню атрибутів, що високо корелюють), а також перетворення методом "просторових знаків" ("spatial sign transformation"), яке буває особливо корисним при наявності багатовимірних викидів.

Бібліотека CARET реалізує близько 200 алгоритмів машинного навчання, які вирішують задачі класифікації та регресії даних. Серед прикладів можна навести лінійний дискримінантний аналіз, наївну байєсову класифікацію, метод опорних векторів, класифікаційні та регресійні дерева рішень тощо. Список всіх методів, що підтримуються бібліотекою, можна знайти на офіційному сайті бібліотеки (https://topepo.github.io/caret/available-models.html) або

виконати код

```
> names(getModelInfo())
```

Всі ці методи навчання моделей застосовуються з використанням однакового синтаксису. Головна функція бібліотеки – це train(), яка використовується для навчання моделей. Ця функція одночасно виконує навчання моделей, знаходження оптимальних параметрів і оцінку якості передбачення моделей. Останні два завдання реалізуються з використанням різноманітних методів створення повторних вибірок (бутстреп, багаторазовий бутстреп, кілька видів перехресної перевірки). Параметри процесу навчання задаються аргументом trControl, на який подається список з параметрами, попередньо створений за допомогою функції trainControl(). Оцінка якості моделі виконується на основі критерію, що задається аргументом metric (зокрема, загальна вірність класифікації "Accuracy", каппа-статистика "Kappa", квадратний корінь з середньоквадратичної помилки "RMSE" і коефіцієнт детермінації "Rsquared"). Алгоритм, за допомогою якого відбувається навчання моделі, вказується за допомогою аргументу method.

В найпростішому випадку навчання моделі виконується наступним чином:

```
> set.seed(123)
> mdl_oner <-train(trainSet[,1:4], trainSet[,5], method='OneR')
```

В цьому прикладі trainSet[,1:4] – це стовпці, що містять незалежні атрибути або іншими словами предиктори, а], trainSet[,5] – це стовпець, що містить цільовий атрибут, тобто клас, або іншими словами відгук.

Ще однією формою запису параметрів навчання моделі є

```
> mdl_oner <-train(Species~., data=trainSet, method='OneR')
```

яка визначає, що буде прогнозуватися атрибут Species на основі всіх інших атрибутів вибірки даних під назвою trainSet.

Результат навчання зберігається в об’єкті. Для того щоб побачити загальні результати навчання можна вивести

```
> mdl_oner
Single Rule Classification

114 samples
 4 predictor
 3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 114, 114, 114, 114, 114, 114, ...
Resampling results:

 Accuracy   Kappa
 0.9547403  0.931186
```

Більш детальну інформацію можна отримати, переглянувши атрибути цього об’єкту.

```
> attributes(mdl_oner)
$names
 [1] "method"     "modelInfo"  "modelType"  "results"    "pred"
 [6] "bestTune"   "call"       "dots"       "metric"     "control"
[11] "finalModel" "preProcess" "trainingData" "resample"
"resampledCM"
[16] "perfNames"  "maximize"   "yLimits"    "times"      "levels"
[21] "terms"      "coefnames"  "xlevels"

$class
[1] "train"       "train.formula"
```

Результуючу модель можна побачити, звернувшись до атрибуту finalModel:

```
> mdl_oner$finalModel
Petal.Width:
  < 0.75   -> setosa
  < 1.75   -> versicolor
  >= 1.75  -> virginica
(111/114 instances correct)
```

Для отримання незміщених оцінок якостей побудованих класифікаційних моделей застосовуються методи формування

повторних вибірок (resampling), за допомогою яких навчальні дані випадковим чином розбиваються на кілька частин. Ці частини потім по черзі використовуються для підгонки різних версій однієї і тієї ж моделі. Кожен раз дані, які не беруть участі в налаштуванні конкретної версії моделі, використовуються для розрахунку відповідного критерію якості. Отримані таким чином значення цього критерію далі усереднюються, що дозволяє отримати незміщену оцінку.

Для налаштування методу створення вибірок під час навчання використовується функція trainControl(). Підтримуються наступні методи створення вибірок:

- boot = bootstrapping
- boot632 = bootstrapping with adjustments
- cv = cross validation
- repeated cv
- loocv = leave one out cv.

Ці налаштування необхідно передати в метод навчання в якості параметру trControl.

Для проведення перехресної перевірки необхідно налаштувати параметри навчання за допомогою функції trainControl() наступним чином:

```
> fitControl <- trainControl(method = "cv", number = 10)
> model <- train(Species~., data=trainSet, trControl=fitControl,
method="nb")
```

Процес пошуку оптимальних значень параметрів моделей з використанням функції train() в загальному вигляді реалізується наступним чином:

1. Визначити набори значень параметрів моделі для оцінки
2. Цикл для кожного набору параметрів моделі:
3. | Цикл для кожної ітерації підвибірки:
4. | | Виділити частину даних для тестування
5. | | Виконати за необхідністю попередню обробку підвибірок
6. | | Підгонка моделі з навчальних об'єктів
7. | | Прогнозування відгуку для тестових об'єктів
8. | end
9. | Обчислення показників середньої ефективності прогнозу

10. end

11. Визначення оптимальних параметрів моделі

12. Підгонка підсумкової моделі по всій вибірці з використанням оптимальних параметрів

Для можливості повторення результатів дослідження не забувайте перед кожним викликом функції train() задати початкове число для генератора випадкових чисел.

Кожний метод навчання має власні параметри налаштування. Щоб визначити, які параметри конкретної моделі можна налаштувати необхідно скористатися функцією modelLookup(model='OneR'). Ці налаштування передаються в метод навчання в якості параметру tuneGrid.

Якщо параметри моделі відомі заздалегідь, параметр tuneGrid має містити таблицю з параметрами налаштування, а метод виконання підвибірок повинен бути вимкнутий, тобто trainControl(method = "none").

Якщо ж потрібно задати можливих значень параметрів для порівняння, необхідно скористатися функцією expand.grid().

```
> grid <- expand.grid(size=c(5,10,20,50), k=c(1,2,3,4,5))
> model <- train(Species~., data=trainSet, method="lvq", trControl=
fitControl, tuneGrid=grid)
```

Якщо параметри налаштування метода не задавати, бібліотека використає по 3 випадкові значення кожного параметру, що може бути налаштовано, і шляхом повторної підвибірки вибере найліпший варіант з найліпшою метрикою (наприклад, найбільшою точністю побудованої моделі).

Для функції train() за допомогою аргументу preProcess можна вказати необхідні параметри попередньої обробки даних.

Бібліотека caret також виконує оцінку важливість атрибутів в кожній моделі за допомогою функції varImp(). Ця функція розраховує кількісні показники, що відображають внесок, або важливість, кожного атрибуту (variable importance) при отриманні прогнозів на основі тієї чи іншої моделі

Функція predict.train() дозволяє отримувати передбачення значень змінної-відгуку на основі нових тестових даних. В якості параметрів виклику необхідно указати ім’я моделі та тестові дані. Для

завдання класифікації, задається додатковий параметр type, який може приймати значення “raw” або “prob”. В першому випадку будуть прогнозуватися лише мітки класу, в другому – будуть прогнозуватися ймовірності приналежності кожного об’єкту до класів.

```
> predictions<-predict.train(object=mdl_oner,testSet[,1:4])
> table(predictions)
predictions
   setosa versicolor  virginica
       12         15          9
```

При роботі з моделями-класифікаторами корисним способом узагальнення якості їх прогнозів є матриця помилок (confusion matrix). Функція confusionMatrix() дозволяє створювати подібні матриці, а також розраховувати цілий набір підсумкових статистик, які характеризують якість прогнозів.

```
> confusionMatrix(predictions,testSet[,"Species"])
Confusion Matrix and Statistics

          Reference
Prediction   setosa versicolor virginica
  setosa         12          0         0
  versicolor      0         12         3
  virginica       0          0         9

Overall Statistics

                Accuracy : 0.9167
                  95% CI : (0.7753, 0.9825)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 3.978e-13

                   Kappa : 0.875
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           0.7500
Specificity                 1.0000            0.8750           1.0000
Pos Pred Value              1.0000            0.8000           1.0000
Neg Pred Value              1.0000            1.0000           0.8889
```

Prevalence	0.3333	0.3333	0.3333
Detection Rate	0.3333	0.3333	0.2500
Detection Prevalence	0.3333	0.4167	0.2500
Balanced Accuracy	1.0000	0.9375	0.8750

Для порівняння якості декількох моделей можна скористатися наступним кодом

```
> # summarize accuracy of models
> results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn,
svm=fit.svm, rf=fit.rf))
> summary(results)
> # compare accuracy of models
> dotplot(results)
```

Завдання

1. Обрати один із вбудованих в середовище мови R набір даних, в якому вирішується задача класифікації.
4. Описати практичну задачу, що описує обраний набір. Описати атрибути та кількість об'єктів в вибірці. Чи є в даних цільовий атрибут?
5. Вирішити задачу класифікації даних за допомогою розглянутих на лекції методів: OneR, наївна Байєсова класифікація, методи побудови дерев рішень CART та C4.5 (або C5.0), метод опорних векторів, метод k найближчих сусідів.
6. Порівняти якість побудованих моделей.

Контрольні питання

1. У чому полягає задача класифікації? Наведіть практичний приклад.
2. Опишіть один із розглянутих методів класифікації.
3. Що таке навчання з учителем і без учителя? До якого типу належить задача класифікації?
4. Задача класифікації є описовою або прогнозуючою і чому?
5. Навіщо потрібні дві вибірки: навчальна і тестова?
6. Які існують підходи для поділу вихідної вибірки на навчальну і тестову?
7. Як оцінити якість побудованої моделі класифікації?

8. Що таке матриця помилок? Як її інтерпретувати?

9. Що означають параметри чутливість, специфічність, точність? Як їх розрахувати?

10. Що таке параметр Каппа? Що він показує?

11. Що таке аналіз втрати-виграші? Навіщо він?

12. Як порівняти роботу двох класифікаторів?

Практична робота №4
Кластерний аналіз даних за допомогою мови R

Мета роботи

Навчитися використовувати мову R для вирішення задачі кластеризації даних.

Теоретичні відомості

Задача кластеризації полягає в розбитті множини об’єктів на групи за їхньою схожістю за ознаками. Вона відноситься до навчання без учителя.

Схожість об’єктів найчастіше визначається за допомогою **метрик відстані**, серед яких найпопулярнішими є Евклідова та Мангеттенська відстані. Ще одним видом метрик, за допомогою яких можна вирахувати схожість об’єктів, є **відстані на основі кореляції**. Найчастіше для розрахунків використовується коефіцієнт кореляції Пірсона.

Для розрахунку відстані між парами об’єктів в мові R є декілька функцій:

– функція dist(), яка є базовою, приймає лише числові дані;

– функція get_dist() пакету factoextra також приймає лише числові дані; на відміну від стандартної функції підтримує метрики засновані на кореляції даних;

– функція daisy() пакету cluster приймає дані будь-яких типів.

Використаємо Евклідову відстань для розрахунку схожості об’єктів вибірки:

```
> dist.eucl <- dist(iris[,1:4], method = "euclidean")
```

Перед вирішенням задачі кластеризації, а саме перед знаходженням відстані між об’єктами даних, може знадобитися видалити об’єкти з невідомими значення ознак або заповнити пропуски в даних. Крім того може знадобитися **нормалізувати** значення ознак у вибірці даних.

```
> mydata <- na.omit(iris) #видалити об’єкти з пропусками
> mydata <- scale(iris) # нормалізувати дані
```

Методи розбиття (partitioning methods) – це алгоритми, які розбивають вибірку даних на задану аналітиком кількість підгруп.

Найбільш популярним та найбільш вживаним є метод k-середніх, в якому кожний кластер подається центром або середнім точок даних, які належать цьому кластеру, і оптимальним є таке розбиття, яке дає найменшу суму квадратів відстаней від точок до кластерів, яким вони належать.

Наведемо приклад вирішення задачі кластеризації методом k-середніх:

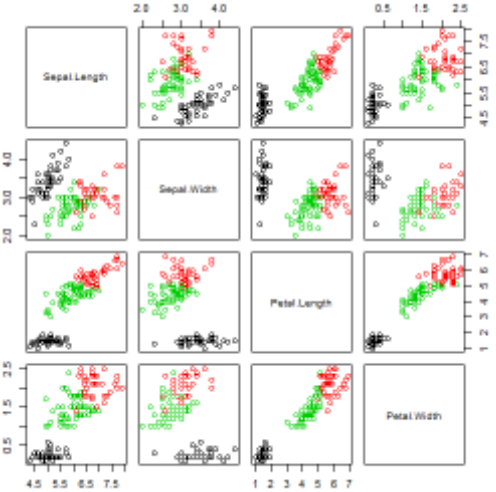
```
> set.seed(123)
> res.km <- kmeans(iris[,1:4], 3, nstart = 20) # 3 кластери
```

Ми розбили дані на три кластери. Так як початкові точки пошуку кластерів є випадковими і від їхнього вибору залежать результати кластеризації, в алгоритмі можна задати кількість ітерацій з різними початковими точками пошуку. В результаті буде обрано результат з найменшою всередині кластерною відстанню.

Результати кластеризації дозволяють переглянути центроїди знайдених кластерів, розподіл об’єктів даних по кластерах і суму відстаней всередині кластерів.

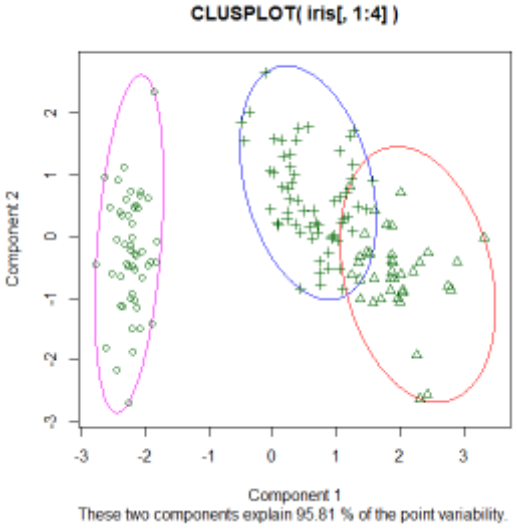
```
> attributes(res.km)
$names
[1] "cluster"    "centers"    "totss"      "withinss"
[5] "tot.withinss" "betweenss"   "size"       "iter"
[9] "ifault"

$class
[1] "kmeans"
> plot(iris[,1:4], col= res.km$cluster)
```



Ще один варіант відображення кластерів

```
> library(cluster)
> clusplot(iris[,1:4], res.km$cluster, color=TRUE, lines=0)
```



Можна порівняти знайдені кластери з класами в даних

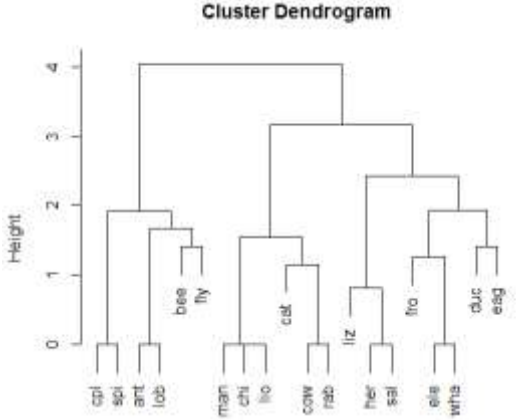
```
> table(res.km$cluster, iris$Species)
   setosa versicolor virginica
1      50         0         0
2       0         2        36
3       0        48        14
```

Більш стійким до шумів в даних є альтернативній алгоритм розбиття навколо медоідів – PAM (Partitioning Around Medoids).

```
> set.seed(123)
> # Compute PAM
> pam.res <- pam(iris[,1:4], 3)
```

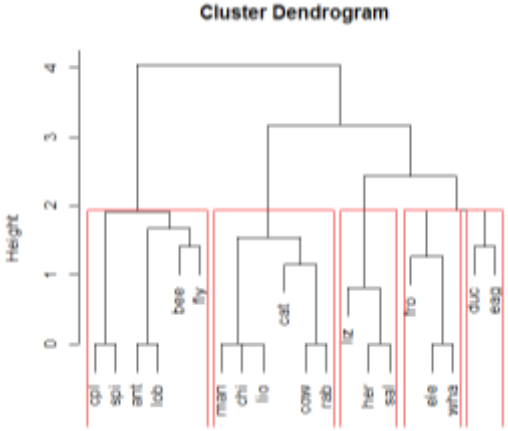
Альтернативним підходом є **ієрархічна кластеризація**. Для неї немає необхідності задавати кількість кластерів, які необхідно знайти. В результаті кластеризації будується деревоподібна структура під назвою дендрограма, що об'єднує об'єкти в вузлах. Об'єкти можуть бути розбиті на кластери шляхом обрізки дендрограми на необхідному рівні.

```
> # Ward Hierarchical Clustering
> d <- dist(animals, method = "euclidean") # distance matrix
> res.hclust <- hclust(d, method="ward.D2")
> plot(res.hclust) # display dendogram
```




```
> groups <- cutree(fit, k=5) # cut tree into 5 clusters
```

```
> # draw dendogram with red borders around the 5 clusters
> rect.hclust(res.hclust, k=5, border="red")
```



Алгоритм ієрархічної кластеризації в якості параметру приймає параметр method, який вказує метод розрахунку відстані між кластерами: “ward.D”, “ward.D2”, “single”, “complete”, “average”, “mcquitty”, “median” or “centroid”.

Для виконання агломеративної та дивізимної кластеризації в мові R передбачено дві функції agnes() та diana().

```
> library("cluster")
> # Agglomerative Nesting (Hierarchical Clustering)
> res.agnes <- agnes(x = animals, stand = TRUE, metric = "euclidean",
method = "ward")
> # DIvisive ANAlysis Clustering
> res.diana <- diana(x = animals,  stand = TRUE,  metric = "euclidean")
```

Нечітка (fuzzy) кластеризація – це такий вид кластеризації, при якому кожний об’єкт належить всім кластерам одночасно з різним ступенем приналежності. При нечіткій кластеризації об’єкти, які знаходяться ближче до центра кластера, будуть мати вищий ступень приналежності кластеру, ніж об’єкти, що знаходяться на краю кластера. При цьому ступінь приналежності об’єкту до кластеру - це числове значення від 0 до 1.

Метод fuzzy c-means – це нечітка модифікація методу k-середніх.

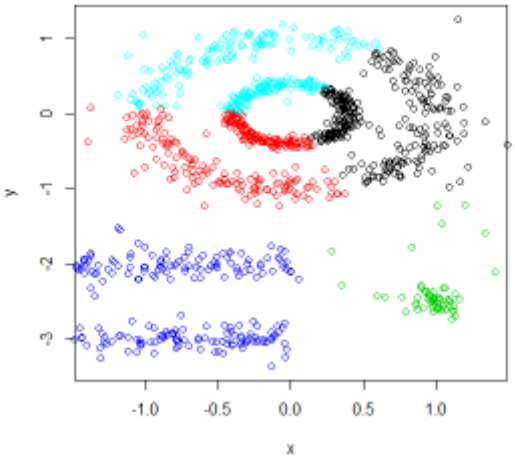
```
> res.fanny <- fanny(iris[, 1:4], 3)  # Compute fuzzy clustering with k=3
> head(res.fanny$membership, 3) # Membership coefficients
          [,1]         [,2]         [,3]
[1,] 0.9142273 0.03603116 0.04974153
[2,] 0.8594576 0.05854637 0.08199602
[3,] 0.8700857 0.05463714 0.07527719
```

Описані вище методи кластеризації є евристичними і не базуються на жодній формальній моделі. Альтернативою такому підходу є кластеризація заснована на **використанні моделей**. Такий вид кластеризації припускає, що дані в кластерах підлягають деякому розподілу. Така кластеризація є нечіткою. Так в методі EM (Expectation-Maximization) кожний з k заданих кластерів моделюється нормальним або Гаусовим розподілом, параметри якого необхідно відшукати.

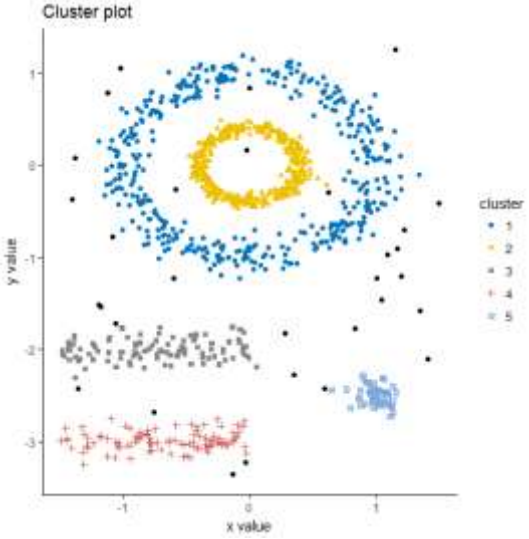
```
> library(mclust)
> res.mc <- Mclust(iris[,1:4])
> plot(res.mc)
```

Метод DBSCAN – це метод кластеризації заснований на **щільності розподілу** точок даних в просторі ознак. Він дозволяє знаходити кластери будь-якої довільної форми в даних з шумами та викидами.

```
> install.packages("fpc")
> install.packages("dbscan")
> install.packages("factoextra")
> # Завантажимо дані для демонстрації
> data("multishapes", package = "factoextra")
> df <- multishapes[, 1:2]
> # Виконаємо кластеризацію методом к-середніх
> set.seed(123)
> km.res <- kmeans(df, 5, nstart = 25)
```




```
> # Виконаємо кластеризацію методом DBSCAN
> library("fpc")
> set.seed(123)
> db <- fpc::dbscan(df, eps = 0.15, MinPts = 5)
> # Відобразимо результати
> library("factoextra")
> fviz_cluster(db, data = df, stand = FALSE,  ellipse = FALSE,
show.clust.cent = FALSE,  geom = "point", palette = "jco", ggtheme =
theme_classic())
```



Пакет factoextra призначений для візуального подання результатів багатомірного аналізу даних. Його функція fviz_cluster призначена для візуалізації результатів кластерного аналізу.

Завдання

1. Обрати один із вбудованих в середовище мови R набір даних.
7. Описати практичну задачу, що описує обраний набір. Описати атрибути та кількість об’єктів в вибірці. Чи є в даних цільовий атрибут?
8. Вирішити задачу кластеризації даних за допомогою розглянутих на лекції методів:.
9. Порівняти якість побудованих моделей.

Контрольні питання

1. У чому полягає задача кластеризації? Наведіть практичний приклад?
2. Що таке навчання з учителем і без учителя? До якого типу належить задача кластеризації?
3. Задача кластеризації є описовою або прогнозуючою і чому?
4. Чим визначається «схожість» об'єктів при вирішенні задачі кластеризації?
5. Що таке однорівнева і ієрархічна кластеризація?
6. Що таке чітка і нечітка кластеризація?
7. Які є підходи до розрахунку відстані між кластерами?
8. Що таке алгомеративна і дівізімна ієрархічна кластеризація?
9. Опишіть один з розглянутих методів, що вирішують завдання кластеризації?
10. Як оцінити якість побудованої моделі для завдання кластеризації?

ЛІТЕРАТУРА

1. https://www.r-project.org/
2. https://www.rstudio.com/
3. А. Б. Шипунов, Е. М. Балдин, П. А. Волкова, А. И. Коробейников, С. А. Назарова, С. В. Петров, В. Г. Суфиянов. Наглядная статистика. Используем R! – 2014. – 296 с.
4. Мастицкий С.Э., Шитиков В.К. (2014) Статистический анализ и визуализация данных с помощью R. – Электронная книга, адрес доступа: http://r-analytics.blogspot.com/p/blog-page_20.html
5. Зарядов Иван Сергеевич. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика. – М.: Изд-во РУДН, 2010. – 207 с.
6. Yanchang Zhao. R and Data Mining: Examples and Case Studies. – 2013. – 160 p.
7. Gareth James,Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning with Applications in R. – Springer, 2013. – 441 p.