

**Міністерство освіти і науки України**

**Харківський національний університет імені В. Н. Каразіна**

**К. Ю. Кононова**

**МАШИННЕ НАВЧАННЯ: МЕТОДИ ТА МОДЕЛІ**

**ПІДРУЧНИК**

**Харків – 2020**

Кононова К. Ю.

УДК 519.2(075.8)

Ч49

**Рецензенти:**

**А. В. Матвійчук** – д.е.н., професор кафедри економіко-математичного моделювання Державний Вищій Навчальний Заклад «Київський національний економічний університет імені Вадима Гетьмана».

**Л. С. Гур'янова** – д.е.н., професор, завідувач кафедри економічної кібернетики Харківського національного економічного університету імені Семена Кузнеця.

*Рекомендовано до друку рішенням Вченої ради  
Харківського національного університету імені В. Н. Каразіна  
(протокол №13 від «23» грудня 2019 р.)*

- Ч49 **Кононова К. Ю. Машинне навчання: методи та моделі:** підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.

У підручнику розглядаються основні принципи аналізу даних та машинного навчання, приділяється увага CRISP-DM методології та питанням підготовки даних. Розглянуто базові моделі лінійної та нелінійної регресії, класифікації (зокрема логістична регресія, методи опорних векторів та k-найближчих сусідів, Байесова класифікація, дерева рішень і випадковий ліс), кластеризації (ієрархічна і k-середніх), а також методи побудови асоціативних правил. Серед більш складних методів машинного навчання розглянуто методи обробки природної мови (модель «Мішок слів», класифікація текстів та аналіз настроїв), використання штучних нейронних мереж в задачах прогнозування, класифікації та кластеризації, подано алгоритми глибокого навчання (згорткові та рекурентні мережі).

Кожен розділ підручника містить теоретичний матеріал, ілюстративні матеріали та лабораторні роботи, виконання яких дозволить студентам оволодіти методами та моделями машинного навчання, а також питання для самоперевірки та завдання для самостійного виконання.

Для бакалаврів, магістрів та докторів філософії, а також викладачів ВНЗ. Підручник буде корисним вченим і фахівцям, які використовують в роботі технології машинного навчання.

УДК 519.2(075.8)

© Харківський національний університет імені В. Н. Каразіна, 2020

© Кононова К. Ю., 2020

© Дончмк І. М., макет обкладинки, 2020

## ЗМІСТ

Вступ.....	5
Розділ 1. Основні поняття інтелектуального аналізу даних .....	8
Тема 1. Завдання аналізу даних .....	8
Лабораторна робота 1 .....	14
Тема 2. CRISP-DM методологія .....	15
Тема 3. Підготовка даних .....	23
Лабораторна робота 2 .....	30
Питання для самоперевірки .....	37
Самостійна робота 1 .....	37
Розділ 2. Регресія.....	38
Тема 4. Лінійна регресія .....	38
Лабораторна робота 3 .....	40
Тема 5. Дерева рішень і випадковий ліс .....	48
Лабораторна робота 4 .....	51
Питання для самоперевірки .....	59
Самостійна робота 2 .....	59
Розділ 3. Класифікація .....	60
Тема 6. Логістична регресія .....	60
Лабораторна робота 5 .....	66
Тема 7. Метод опорних векторів.....	74
Лабораторна робота 6 .....	79
Тема 8. Метод k-найближчих сусідів .....	86
Лабораторна робота 7 .....	88
Тема 9. Байєсова класифікація .....	91
Лабораторна робота 8 .....	95
Тема 10. Дерева рішень та ансамблеві методи.....	99
Лабораторна робота 9 .....	101
Питання для самоперевірки .....	111
Самостійна робота 3 .....	111
Розділ 4. Кластеризація .....	112
Тема 11. Ієрархічна кластеризація .....	112
Лабораторна робота 10 .....	114
Тема 12. Кластеризація на основі k-means .....	118
Лабораторна робота 11 .....	122
Питання для самоперевірки .....	127
Самостійна робота 4 .....	127

Розділ 5. Асоціативні правила .....	128
Тема 13. Побудова асоціативних правил.....	128
Тема 14. Алгоритми APRIORI та ECLAT .....	132
Лабораторна робота 12 .....	139
Питання для самоперевірки.....	145
Самостійна робота 5.....	145
Розділ 6. Обробка природної мови.....	146
Тема 15. Модель «мішок слів» .....	146
Лабораторна робота 13 .....	150
Тема 16. Моделі з урахуванням семантики .....	153
Питання для самоперевірки.....	154
Самостійна робота 6.....	154
Розділ 7. Штучні нейронні мережі .....	155
Тема 17. Теоретичні основи нейронних мереж.....	155
Тема 18. НМ в задачах апроксимації та прогнозування .....	169
А. НМ в задачах апроксимації .....	169
Лабораторна робота 14 .....	171
Б. НМ в задачах прогнозування .....	180
Лабораторна робота 15 .....	180
Тема 19. НМ в задачах класифікації .....	188
Лабораторна робота 16 .....	188
Тема 20. НМ в задачах кластеризації: карти Кохонена .....	194
Лабораторна робота 17 .....	196
Питання для самоперевірки.....	204
Самостійна робота 7.....	204
Розділ 8. Глибоке навчання .....	205
Тема 21. Рекурентні нейронні мережі.....	206
Лабораторна робота 18 .....	210
Тема 22. Згорткові нейронні мережі .....	215
Лабораторна робота 19 .....	219
Питання для самоперевірки.....	221
Самостійна робота 8.....	222
ДОДАТКИ .....	223
Література.....	299

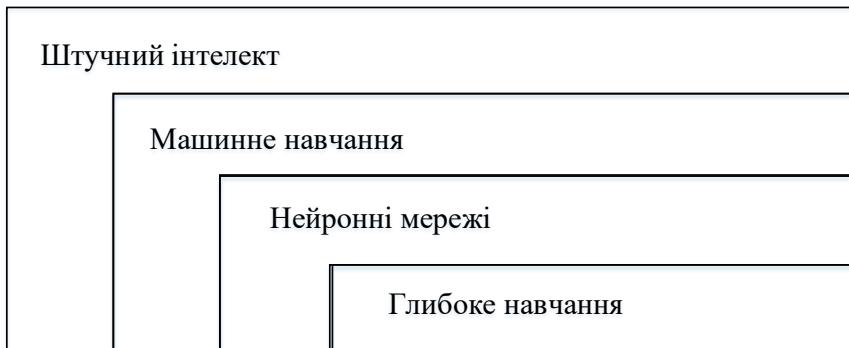
## ВСТУП

Стрімкий розвиток ІТ-галузі в Україні відображає світові тенденції переходу до інформаційної економіки. Запит на фахівців з машинного навчання зростає з року в рік. Обсяги інформації, що генеруються та зберігаються, зростають експоненціально, однак без використання спеціальних засобів аналізу вони не приносять користі. Дуже часто ухвалення рішення ускладнене тим, що хоча дані є, вони неповні або, навпаки, надмірні, зашумлені, не систематизовані або систематизовані не правильно.

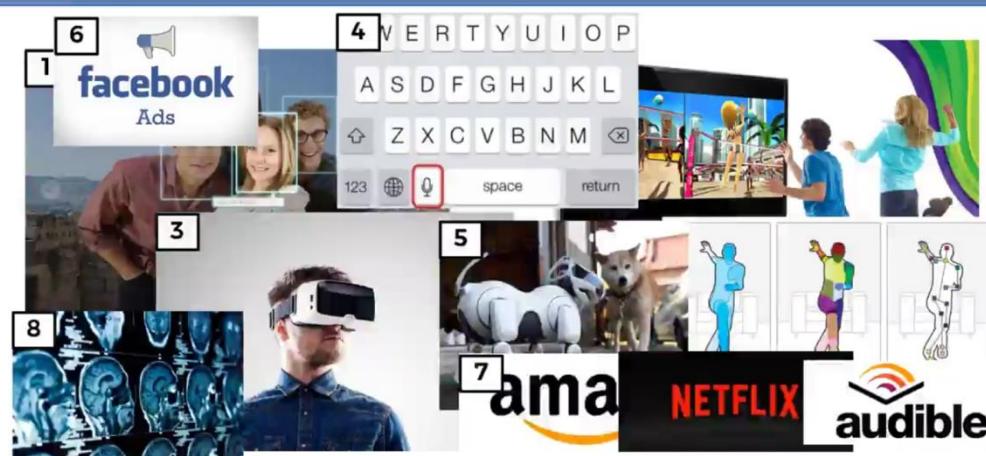
Для подолання цієї проблеми було розроблено різні технології інтелектуального аналізу даних, що дозволяють виймати корисні знання із баз даних. Спочатку ці технології розвивалися в межах напряму data mining (аналіз даних), що являють собою сукупність методів виявлення в «сирих» даних раніше невідомих, нетривіальних, практично корисних і доступних для інтерпретації знань, необхідних для ухвалення рішень в різних сферах людської діяльності.

Термін data mining введений Григорієм Пятецким-Шапіро 1989 року. Сьогодні до методів data mining нерідко відносять статистичні методи (дескриптивний аналіз, кореляційний та регресійний аналіз, факторний аналіз, дисперсійний аналіз, компонентний аналіз, дискримінантний аналіз, аналіз часових рядів). Проте такі методи припускають деякі априорні уявлення про аналізовані дані, тому виникає певна розбіжність з цілями data mining (виявлення раніше невідомих нетривіальних і практично корисних знань).

З плином часу та розвитком галузі машинне навчання виділилося в окремий мультидисциплінарний напрямок. Особливістю методів машинного навчання є не прямий розв'язок задачі, а навчання на множині подібних прикладів, що дозволяє використовувати ці методи для обробки великих обсягів даних та виявляти в них нові, нетривіальні, корисні та доступні для інтерпретації знання. Okрім методів штучного інтелекту, під час розробки моделей машинного навчання в якості допоміжних використовуються засоби математичної статистики, чисельних методів, методів оптимізації, теорії ймовірностей, теорії графів, різні техніки роботи з даними в цифровій формі.



Технології аналізу даних і машинного навчання використовуються для розпізнавання зображень, розробки додатків доповненої реальності і комп'ютерного зору, текстового запису з голосу, в робототехніці, для таргетування реклами та створення системи рекомендацій тощо.



Підручник «Машинне навчання: методи та моделі» розроблено для підготовки бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка». У підручнику розглянуто теми, що входять до навчальних дисциплін «Інтелектуальні системи аналізу даних» та «Сучасні технології моделювання економічних процесів» освітньо-професійних програм «Економічна кібернетика» та «Прикладна економіка».

У підручнику розглядаються 22 теми, які об'єднано в 8 розділів: «Основні поняття інтелектуального аналізу даних», «Регресія», «Класифікація», «Кластеризація», «Асоціативні правила», «Обробка природної мови», «Штучні нейронні мережі», «Глибоке навчання». Кожна тема містить теоретичний матеріал, лабораторні роботи із скриптами програмної мови R, коментарями до них і результатами виконання, а також питаннями для самоперевірки та завданнями для самостійної роботи.

Список рекомендованої літератури охоплює фундаментальні наукові праці провідних авторів і навчальну літературу по машинному навчанню.

Підручник може бути рекомендований для студентів денної, заочної та дистанційної форм навчання, зокрема й у разі кредитно-модульної системи організації навчального процесу.

Автор висловлює глибоку подяку д. е. н., професору А. В. Матвійчуку, д. е. н., професору Л. С. Гур'яновій за рецензування цієї роботи, а також керівництву Харківського національного університету імені В. Н. Каразіна та економічного факультету за сприяння в підготовці та виданні підручника.

## РОЗДІЛ 1. ОСНОВНІ ПОНЯТТЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

### **Тема 1. Завдання аналізу даних**

Методи машинного навчання використовуються для вирішення таких основних завдань: прогнозування, класифікація та кластеризація, побудова асоціативних правил.

**1. Прогнозування** – встановлення функціональної залежності між вхідними і неперервними вихідними змінними. Прогнозування найчастіше зводиться до розв'язку задачі регресії.

*Регресія, зокрема, використовується під час прогнозування обсягів продажів, в цьому випадку залежною величиною є обсяги продажів, а факторами, що на неї впливають, можуть бути попередні обсяги продажів, зміна курсу валют, активність конкурентів тощо.*

**2. Класифікація** – встановлення функціональної залежності між вхідними і дискретними вихідними змінними. За допомогою класифікації вирішується завдання приналежності об'єктів до одного зі заздалегідь відомих класів.

*Наприклад, приналежність нового товару до тієї чи іншої товарної групи або клієнта до будь-якої категорії. Під час кредитування це може бути належність клієнта за якимсь ознаками до однієї з груп ризику.*

**3. Кластеризація** – групування об'єктів на основі їх властивостей. Об'єкти в середині кластера повинні бути схожими і відрізнятися від об'єктів, що увійшли в інші кластери. Чим більше схожі об'єкти в середині кластера і чим більше відмінностей між кластерами, тим точніше кластеризація.

*Наприклад, кластеризація може використовуватися для сегментування і побудови профілів покупців. За досить великої кількості клієнтів стає важко підходити до кожного індивідуально. Тому їх зручно об'єднати в групи – сегменти з подібними ознаками. Після кластеризації можна дізнатися, які саме групи є найбільш активними, які найбільш прибуткові, виділити характерні для них ознаки. Ефективність роботи з клієнтами підвищується за рахунок обліку їх персональних або групових переваг.*

**4. Асоціація** – виявлення залежностей між пов’язаними подіями, що вказують, що з події  $X$  випливає подія  $Y$ . Такі правила називаються асоціативними. Вперше ця задача була запропонована для знаходження типових шаблонів покупок, що здійснюються в супермаркетах, тому іноді її називають аналізом споживчого кошика (market basket analysis).

*Асоціації допомагають виявляти спільно придбані товари. Це може бути корисно для більш зручного розміщення товару на прилавках, стимулювання продажів.*

Перераховані вище базові методи аналізу даних слугують для створення аналітичних систем з використанням, зокрема, таких алгоритмів машинного навчання:



Ринок програмного забезпечення, що підтримує методи й алгоритми інтелектуального аналізу даних, досить широкий і представлений, наприклад, такими продуктами: R, Python, SPSS, Statsoft Statistica, SAS Enterprise Miner, Oracle DM, MATLAB, Viscovery, Orange, WEKA, Deductor. В межах курсу для вирішення завдань аналізу даних будемо використовувати R.

R – статистична система аналізу, створена Россом Ихак і Робертом Гентлеманом. R є і мовою, і програмним забезпеченням. Його особливості:

- ефективна обробка даних і прості засоби для збереження результатів;
- набір операторів для обробки масивів, матриць та інших складних конструкцій;
- велика, послідовна, інтегрована колекція інструментальних засобів для проведення статистичного аналізу;
- численні графічні засоби;

– проста та ефективна мова програмування, що містить багато можливостей.

R доступний в декількох формах: вихідний текст програм, написаний на C і у відкомпільованому вигляді. R – мова з багатьма функціями для виконання статистичного аналізу та графічного відображення результатів, які візуалізуються відразу у власному вікні та можуть бути збережені в різних форматах (наприклад, jpg, png, bmp, eps, або wmf під Windows, ps, bmp, pictex під Unix).

Результати статистичного аналізу можуть бути відображені на екрані. Деякі проміжні результати (P-values, коефіцієнт регресії тощо) можуть бути збережені в файлі та використовуватися для подальшого аналізу.

R – мова, що дозволяє користувачеві використовувати оператори циклів, щоб послідовно аналізувати кілька сукупностей даних. Також можливо об'єднати в окрему програму різні статистичні функції для проведення більш складного аналізу.

Інстальована система R запускається викликом файлу R Studio. Підказка «>» вказує, що R чекає ваших команд. Деякі команди, пов’язані з операційною системою (доступ до довідки, відкриття файлів), можуть бути виконані за допомогою меню, але більшість необхідно набирати на клавіатурі.

R – об’єктно-орієнтована мова: змінні, дані, матриці, функції, результати тощо зберігаються в оперативній пам’яті комп’ютера в формі об’єктів, які мають назvu. Для відображення його значення необхідно надрукувати назvu об’єкта. Наприклад, для відображення значення об’єкта n

```
> n
```

```
[1] 10
```

Цифра 1 в дужках вказує, що відображається перший елемент n, що має значення 10.

Для того, щоб привласнити значення об'єкта використовується символ «`<-`». Цей символ пишеться разом зі знаком мінус так, щоб вони являли собою стрілку, що може бути спрямована зліва направо, або навпаки

```
> n <- 15
```

```
> n
```

```
[1] 15
```

```
> 5 -> n
```

```
> n
```

```
[1] 5
```

Значення також може бути результатом арифметичного виразу

```
> n <- 10 + 2
```

```
> n
```

```
[1] 12
```

Можна просто надрукувати вираз, не привласнюючи йому назву, тоді результат буде відображенний на екрані, але не збережеться у пам'яті

```
> (10 + 2) * 5
```

```
[1] 60
```

Р працює з об'єктами, що мають два вбудовані атрибути: тип даних і довжина. Тип даних – вид елемента; є чотири типи даних:

- num (числовий);
- char (символьний);
- complex (комплексний);
- logical (логічний).

Довжина – загальна кількість елементів об'єкта.

Наведемо список можливих типів даних для різних об'єктів:

- vector (вектор) – звичайна змінна;
- factor (фактор) – категорійна змінна;
- array (масив) – таблиця з  $k$  вимірами;

- `matrix` (матриця) – окремий випадок масиву з  $k = 2$ . У масиві та матриці всі елементи одного і того ж типу;
- `Data.frame` – таблиця складається з декількох векторів однакової довжини, але можливо різних типів;
- `Ts` – набір послідовних даних, що містить додаткові атрибути, такі як частота та дата.

Серед невластивих атрибутів об'єкта може бути `dim` (розмірність), що відповідає вимірам багатовимірного об'єкта. Наприклад, матриця з 2 рядками і 2 стовпцями має для `dim` пару значень [2,2], але його довжина – 4.

R розрізняє в назвах об'єктів великі і малі літери.

Перелік основних функцій та довідкові матеріали щодо окремих бібліотек наведено в додатках.

## Лабораторна робота 1

### R INSTALLATION

Завантажте та встановіть R – <https://www.r-project.org/> (або <https://cran.r-project.org/>) і середовище розробки R Studio – <https://www.rstudio.com>.

Запустіть RStudio. В правій нижній частині екрана оберіть вкладку Packages і, натиснувши кнопку Install, встановіть потрібні пакети (це можна робити і пізніше в міру потреби).

Бібліотеки за замовчуванням:

`library (base)`, `library(datasets)`, `library (graphics)`,  
`library(grDevices)`, `library (methods)`, `library(stats)`, `library (utils)`

Бібліотеки, що потребують завантаження:

*Підготовка даних*

`library (dplyr)`, `library(ggplot2)`, `library (psych)`,  
`library(caTools)`

*Регресія*

`library(rpart)`, `library(randomForest)`

*Класифікація*

`library(ROCR)`, `library(ElemStatLearn)`, `library(e1071)`,  
`library(class)`

*Кластеризація*

`library(cluster)`

*Асоціативні правила*

`library(arules)`, `library(arulesViz)`

*Обробка природної мови*

`library(tm)`, `library(SnowballC)`

*Штучні нейронні мережі*

`library(nnet)`, `library(graphics)`, `library(neuralnet)`,  
`library(kohonen)`, `library(forecast)`

*Глибоке навчання*

`library(keras)`, `library(tensorflow)`

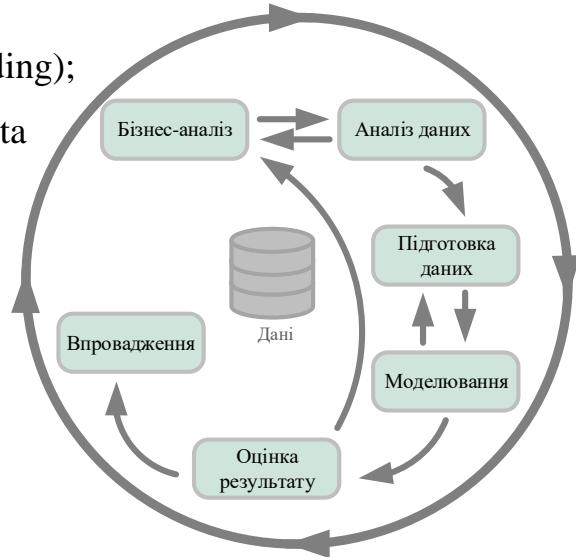
## Тема 2. CRISP-DM методологія

Cross Industry Standard Process for Data Mining (CRISP-DM) – стандарт, що описує загальні процеси й підходи до аналітики даних, що використовуються в промислових data-mining проектах незалежно від конкретного завдання й індустрії.

Методологія, розроблена 1996 року за ініціативою трьох компаній (Daimler Chrysler, SPSS і Teradata), далі допрацьовувалася за участю понад 200 компаній різних індустрій. Всі ці компанії використовували різні аналітичні інструменти, але процес у всіх був побудований схоже.

Згідно із CRISP-DM, аналітичний проект складається з шести основних етапів:

1. Бізнес-аналіз (Business understanding);
2. Первінний аналіз даних (Data understanding);
3. Підготовка даних (Data preparation);
4. Моделювання (Modeling);
5. Оцінка результату (Evaluation);
6. Впровадження (Deployment).



Методологія не жорстка. Вона допускає варіацію залежно від конкретного проекту – можна повернутися до попередніх кроків, якісь кроки можна пропускати, якщо для розв’язуваної задачі вони не важливі.

Кожен з цих етапів зі свого боку поділяється на завдання. На виході кожного завдання повинен виходити певний результат.

Розглянемо перераховані етапи докладніше.

## **1. Бізнес-аналіз (Business Understanding)**

### ***1.1. Мета проекту (Business objectives)***

Під час формульовання мети проекту необхідно проаналізувати:

- організаційну структуру проекту: список учасників з боку замовника, джерела фінансування, користувачів;
- бізнес-мету проекту;
- вже розроблені рішення і те, що в них не влаштовує.

### ***1.2. Поточна ситуація (Assessing current solution)***

Варто оцінити ресурси проекту:

- чи є доступне hardware або його необхідно закуповувати;
- де і як зберігаються дані, чи буде надано доступ до цих систем, чи потрібно додатково збирати зовнішні дані;
- чи зможе замовник виділити своїх експертів для консультацій на цей проект.

Також потрібно описати ймовірні ризики проекту, а також визначити план дій щодо їх зменшення, а саме:

- злив термінів;
- фінансові ризики (наприклад, спонсор втратить зацікавленість в проекті);
- мала кількість або погана якість даних, що не дозволять побудувати ефективну модель;
- дані якісні, але закономірностей немає, тому отримані результати не цікаві замовнику.

### ***1.3. Завдання, які вирішуються з точки зору аналітики (Data Mining goals)***

Після того, як завдання сформульоване в бізнес-термінах, необхідно описати його в технічних термінах. Зокрема, відповісти на такі питання:

- яка метрика буде використовуватися для оцінки результату моделювання;
- який критерій успішності моделі;
- якщо об'єктивний критерій якості не буде використовуватися, то як будуть оцінюватися результати.

#### **1.4. План проекту (Project Plan)**

Як тільки отримані відповіді на всі основні питання, слід скласти план проекту, що повинен містити оцінку всіх шести фаз впровадження.

### **2. Первинний аналіз даних (Data Understanding)**

Мета кроку – зрозуміти слабкі та сильні боки даних.

#### **2.1. Збір даних (Data collection)**

Спочатку потрібно розуміти, які дані має замовник. Дані можуть бути:

- власні (1<sup>st</sup> party data);
- сторонні дані (3<sup>rd</sup> party);
- «потенційні» дані (для отримання яких необхідно організувати збір).

Необхідно проаналізувати всі джерела, доступ до яких надає замовник. Якщо власних даних недостатньо, можливо, варто придбати сторонні або організувати збір нових даних.

#### **2.2. Опис даних (Data description):**

- необхідно описати дані у всіх джерелах (таблиця, ключ, кількість рядків, кількість стовпців, обсяг на диску);
- якщо обсяг дуже великий для використовуваного ПО, створюється семпл даних;
- розраховуються ключові статистики за атрибутами (мінімум, максимум, розкид тощо).

### **2.3. Дослідження даних (*Data exploration*)**

За допомогою графіків і таблиць потрібно досліджувати дані, щоб сформулювати гіпотези щодо постановки завдання.

### **2.4. Якість даних (*Data quality*)**

До моделювання важливо оцінити якість даних:

- пропущені значення;
- помилки даних (описки);
- неконсистентне кодування значень (наприклад, «м» і «male» в різних системах).

## **3. Підготовка даних (*Data Preparation*)**

Підготовка даних – найбільш витратний за часом етап machine learning проекту (50 – 70 % часу проекту). Мета етапу – підготувати навчальну вибірку для використання в моделюванні.

### **3.1. Відбір даних (*Data Selection*)**

Спочатку потрібно відібрати дані, які будуть використовуватися для навчання моделі. Відбираються як атрибути, так і кейси. Під час вибору даних аналітик відповідає на такі питання:

- яка потенційна релевантність атрибута розв'язуваної задачі;
- чи достатньо якісний атрибут для використання в моделі;
- чи варто додуточнити атрибути, що корелують один з одним;
- чи є обмеження на використання атрибутів.

### **3.2. Очищення даних (*Data Cleaning*)**

Коли потенційно цікаві дані відібрані, потрібно перевірити їх якість:

- пропущені значення потрібно або заповнити, або видалити з розгляду;
- помилки в даних спробувати виправити вручну або видалити з розгляду;
- звести до єдиної системи кодування.

Отримуємо 3 списки атрибутів – якісні, виправлені та браковані атрибути.

### ***3.3. Генерація даних (Constructing new data)***

Генерація ознак (feature engineering) – це найбільш важливий етап в підготовці даних: грамотно складена ознака може істотно поліпшити якість моделі. До генерації даних належить:

- агрегація атрибутів (розрахунок sum, avg, min, max, var тощо);
- генерація кейсів (наприклад, oversampling);
- конвертація типів даних для використання в різних моделях (інтервалльні, номінальні дані);
- нормалізація атрибутів (feature scaling).

### ***3.4. Інтеграція даних (Integrating data)***

Найчастіше дані необхідно завантажувати з декількох джерел, і для підготовки навчальної вибірки потрібна їх інтеграція. Під інтеграцією розуміється як горизонтальне з'єднання (merge), так і вертикальне об'єднання (append), а також агрегація даних. Отримуємо єдину аналітичну таблицю, придатну для роботи в якості навчальної вибірки.

### ***3.5. Форматування даних (Formatting Data)***

На цьому етапі дані потрібно звести до формату, придатного для моделювання.

## **4. Моделювання (Modeling)**

На четвертому етапі проводиться навчання моделей. Зазвичай воно виконується ітераційно – тестиються різні моделі, порівнюються їх якість, здійснюється перебирання гіперпараметрів, обирається найкраща комбінація.

### ***4.1. Вибір алгоритмів (Selecting the modeling technique)***

На цьому етапі необхідно визначитися, які моделі будуть використовуватися. Вибір моделі залежить від розв'язуваної задачі, типів атрибутів і вимог до складності. Під час вибору слід звернути увагу на таке:

- чи достатньо даних (складні моделі зазвичай вимагають більшої вибірки);
- чи зможе модель обробити пропуски даних (не всі реалізації алгоритмів вміють працювати з пропусками);
- чи зможе модель працювати з наявними типами даних або необхідна конвертація.

#### ***4.2. Планування тестування (Generating a test design)***

Традиційний підхід – розподіл вибірки на 3 частини (навчальну, валідаційну та тестову) в пропорції 60/20/20. В цьому випадку навчальна вибірка використовується для оцінки параметрів моделі, а валідаційна і тестова – для оцінки її якості, без ефекту перенавчання<sup>1</sup>. Більш складні стратегії передбачають використання різних варіантів крос-валідації.

#### ***4.3. Навчання моделей (Building the models)***

На цьому етапі запускається цикл навчання, і після кожної ітерації фіксуються результати:

- чи виявлені цікаві закономірності;
- швидкість навчання/застосування моделі;
- чи були проблеми з якістю даних.

#### ***4.4. Оцінка результатів (Assessing the model)***

Після того, як сформований пул моделей, їх потрібно ще раз детально проаналізувати й скласти відсортований список моделей. Завдання кроку:

- провести технічний аналіз якості моделі;
- оцінити, чи готова модель до впровадження;
- чи досягаються задані критерії якості;
- оцінити результати з точки зору досягнення бізнес-цілей.

---

<sup>1</sup> При перенавчанні (англ. overfitting) статистична модель описує випадкову похибку або шум, замість взаємозв'язку, що лежить в основі даних. Перенавчання виникає тоді, коли модель є занадто складною, такою, що має занадто багато параметрів відносно числа спостережень. Перенавчена модель має погану передбачувальну здатність, оскільки вона занадто сильно реагує на другорядні відхилення в тренувальних даних.

Якщо критерій успіху не досягнуто, то можна, або покращувати поточну модель, або спробувати нову. Перш ніж переходити до впровадження потрібно переконатися, що:

- результат моделювання зрозумілий і логічний;
- випробувані всі доступні моделі;
- інфраструктура готова до впровадження моделі.

## 5. Оцінка результату (Evaluation)

### 5.1. Оцінка результатів моделювання (Evaluating the results)

Якщо на попередньому етапі результати моделювання оцінювалися з технічної точки зору, то тут – з точки зору досягнення бізнес-цілей:

- слід сформулювати результат у бізнес-термінах (в \$ і ROI, а не Lift або R<sup>2</sup>);
- оцінити, наскільки ефективно отримані результати вирішують бізнес-завдання.

### 5.2. Аналіз процесу виконання проекту (Review the process):

- чи можна було якісь кроки зробити більш ефективними;
- які були допущені помилки, як їх уникнути в майбутньому;
- чи були гіпотези, що не спрацювали, чи варто їх перевіряти повторно;
- чи були несподіванки під час реалізації кроків, як їх передбачити в майбутньому.

### 5.3. Ухвалення рішення (Determining the next steps)

Далі потрібно або впроваджувати модель, якщо вона влаштовує замовника, або, якщо видно потенціал для поліпшення, спробувати ще її поліпшити.

## **6. Впровадження (Deployment)**

На цьому етапі здійснюється впровадження моделі, під яким розуміється як фізичне додавання функціоналу, так і зміни в бізнес-процесах компанії.

### ***6.1. Планування впровадження (Planning Deployment):***

- важливо зафіксувати, що саме і в якому вигляді буде впроваджуватися, а також підготувати технічний план впровадження;
- продумати, як із впроваджуваною моделлю працюватимуть користувачі;
- визначити принцип моніторингу рішення, якщо потрібно, підготуватися до промислової експлуатації.

### ***6.2. Налаштування моніторингу моделі (Planning Monitoring)***

Дуже часто до проекту входять роботи з підтримки рішення:

- які показники якості моделі будуть відслідковуватися;
- як визначити, що модель застаріла;
- якщо модель застаріла, чи достатньо її перенавчити або потрібно організовувати новий проект.

### ***6.3. Звіт за результатами моделювання (Final Report)***

Після закінчення проекту пишеться звіт про результати, починаючи від первинного аналізу даних і закінчуєчи впровадженням моделі. До цього звіту також можна додати рекомендації щодо подальшого розвитку моделі. Звіт презентується замовнику і всім зацікавленим особам. За браком ТЗ цей звіт є головним документом проекту.

### Тема 3. Підготовка даних

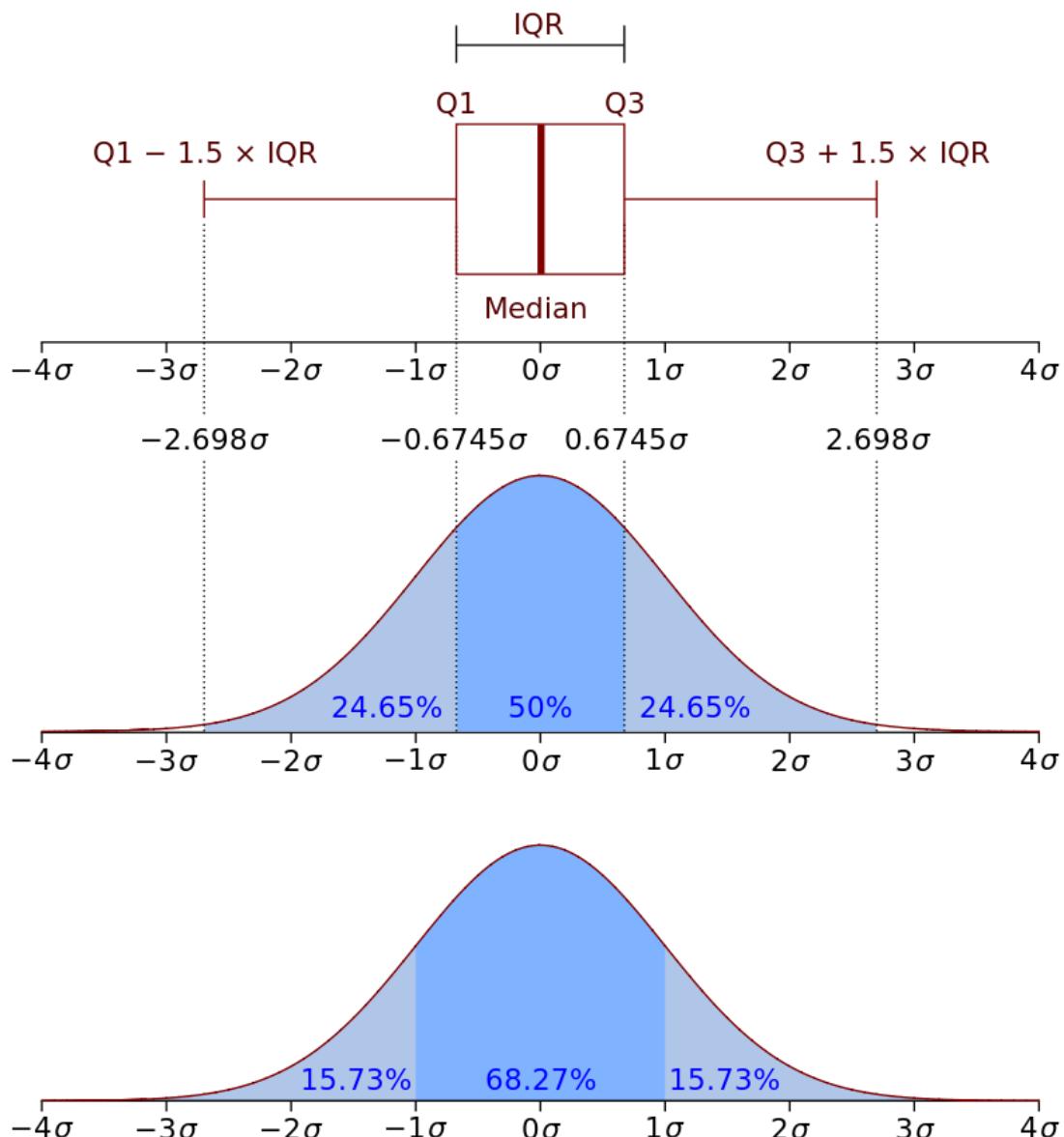
**Кодування.** Економічні дані, з якими ми будемо працювати в рамках цього курсу, можна розбити на два великі класи – числові та категоріальні (номінальні).

Для кодування категоріальних змінних спочатку необхідно скласти перелік усіх можливих значень, а потім провести кодування. Розрізняють просте кодування – відповідно доожної категорії береться якесь числове значення, і розширене кодування – кожній категорії зіставляється бінарна змінна.

Недоліком простого кодування є можливість інтерпретації моделлю категорій, що перебувають в середині списку, як комбінацій категорій з початку та кінця списку (наприклад, якщо категорії *коні* – *корови* – *свині* закодовані порядковими 1 – 2 – 3, то модель буде інтерпретувати корову як середнє між *конем* і *свинею*). Крім того, не виключена ймовірність отримання безглазих неціличесельних відповідей. Недолік розширеного кодування полягає в збільшенні кількості змінних моделі.

**Обчислення статистик.** Для побудови добре збалансованої моделі важливо, щоб розподіл тренувальних даних був гладким. Основна маса економічних даних може бути описана нормальним законом розподілу, що і необхідно насамперед перевірити. Для неперервних даних повинні бути обчислені: середнє значення, стандартне відхилення, максимум і мінімум.

**Видалення викидів.** Дев'яносто п'ять відсотків даних, розподілених за нормальним законом, перебувають всередині інтервалу, обмеженого подвоєним значенням стандартного відхилення в околиці середнього значення (три стандартні відхилення охоплюють 99 % даних).



Величини, занадто далекі від середнього, можуть вплинути на якість моделі. Найпростіше рішення цієї проблеми – видалення викидів, тобто виключення з розгляду даних, що перебувають поза зазначенним інтервалом.

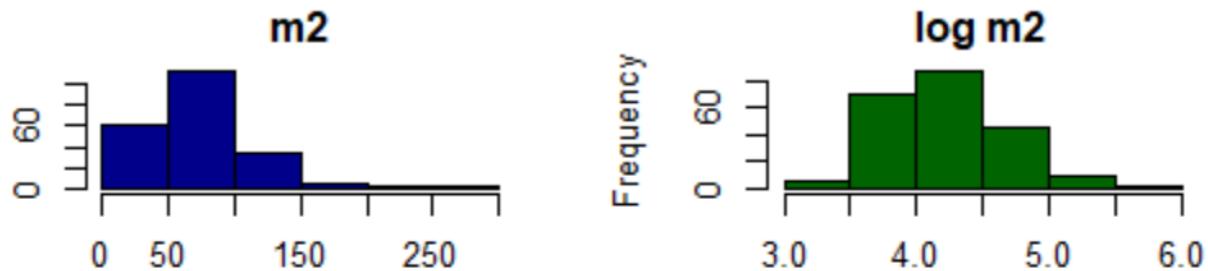
Недоліком такої методики є те, що вона працює для кожної зі змінних незалежно, в той час як викид може тлумачитися як неприйнятна комбінація величин. Вони, якщо беруться окремо, не виходять за межі заданого критерію, але в комбінації перебувають за межами хмари нормально розподілених даних.

Якщо даних мало, то екстремальні величини можуть бути замінені, наприклад, своїми середніми значеннями.

**Обробка помилок та відсутніх значень.** Одним з найпростіших методів обробки помилок та відсутніх значень є видалення відповідних записів. Але зазвичай цей метод не оптимальний. В таких випадках:

- можливо використовувати найбільш ймовірне значення ознаки (середнє або медіану для дійсних змінних, найчастіше для категоріальних);
- для упорядкованих даних (наприклад, часових рядів) можна брати сусіднє значення – наступне або попереднє.

**Шкалювання даних.** В разі асиметричних даних використовується додавання або віднімання константи.



Рівним відстаням на логарифмічній шкалі відповідають рівні процентні збільшення на вихідній шкалі. При цьому:

- якщо беруться логарифми ендогенної ( $Y$ ) та екзогенної ( $X$ ) змінних, то коефіцієнт при факторній змінній ( $b_i$ ) відповідає еластичності (на скільки відсотків зміниться  $Y$  при зміні  $X$  на 1%),
- якщо логарифмуванню піддається тільки  $Y$ , то  $b_i$  показує, що при зміні  $X$  на 1 одиницю,  $Y$  зміниться на 1%,
- у зворотному випадку  $b_i$  свідчить, що при зміні  $X$  на 1%,  $Y$  зміниться на  $b_i / 100$  одиниць.

Іноді одна змінна змінюється в діапазоні, наприклад, від 10 000 до 10 000 000, а інша в діапазоні від 0,3 до 0,6. В цьому разі помилки, зумовлені впливом першої змінної, сильніше впливатимуть на навчання, ніж помилки, обумовлені впливом другої, що змінюється у вузьких межах. Шкалювання всіх змінних в один діапазон забезпечує рівний вплив кожної змінної.

Найчастіше використовують такі методи шкалювання:

- 1) стандартизація (Standart Scaling):

$$\tilde{x} = \frac{x - \bar{x}}{\sigma}$$

- 2) нормалізація відносно мінімуму (MinMax Scaling):

$$\tilde{x} = \frac{x - \min\{x_i\}}{\max\{x_i\} - \min\{x_i\}}$$

- 3) нормалізація відносно середнього (Avg Scaling):

$$\tilde{x} = \frac{x - \bar{x}}{\max\{x_i\} - \min\{x_i\}}$$

Перевагою шкалювання є збереження співвідношення між величинами, воно добре працює, якщо дані розподілені рівномірно. Застосування шкалювання до даних, які розподілені нерівномірно або містять викиди, призводить до того, що дані виявляються розподіленими в дуже вузькому діапазоні.

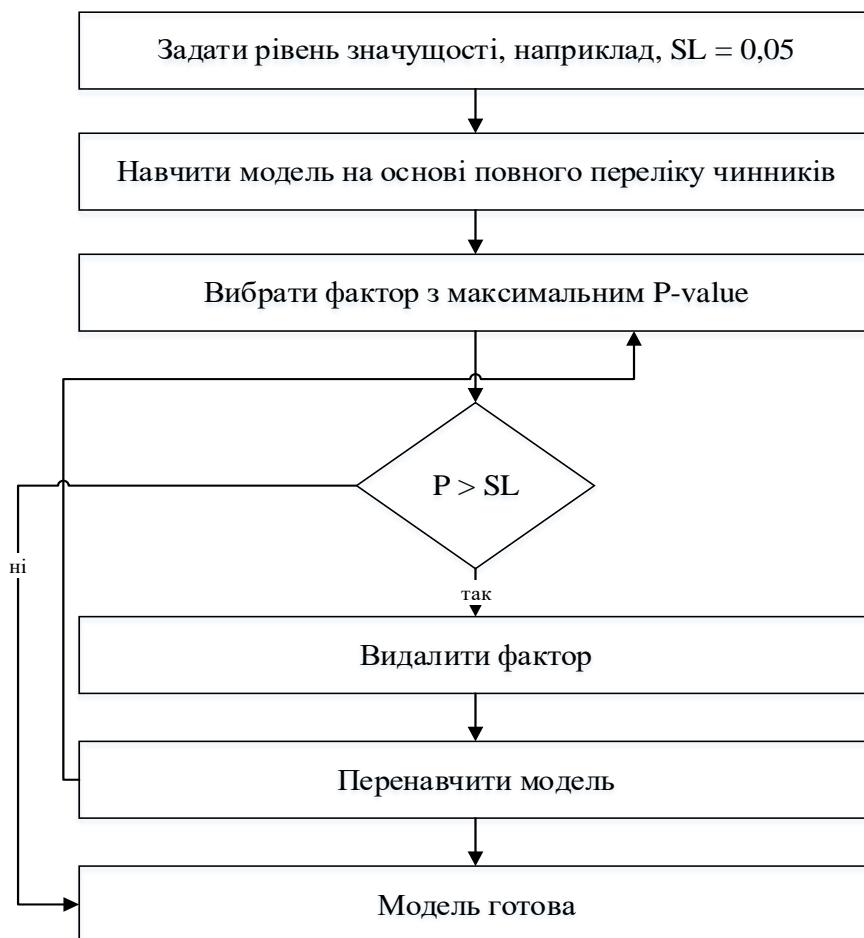
Розподіли з важким правим хвостом можуть підлягати логарифмуванню. Також можна застосовувати перетворення Бокса-Кокса (Box-Cox transformation) (логарифмування – окремий випадок трансформації Бокса-Кокса) або перетворення Йео-Джонсона (Yeo-Johnson), що розширює сферу застосування на негативні числа.

**Вибір ознак (Feature selection).** Є мінімум дві важливі причини позбавлятися від непотрібних ознак:

- 1) чим більше даних, тим вище обчислювальна складність;
- 2) деякі алгоритми тлумачать шум (неінформативні ознаки) як сигнал, що призводить до перенавчання моделі.

Статистичний підхід до відсіву змінних полягає у видаленні змінних, дисперсія яких нижче певної границі.

Також можна використовувати базову модель для оцінки ознак, на основі якої можна оцінити значущість ознак: якщо ознаки не корисні в простій моделі, не потрібно використовувати їх і в більш складній.



Backward Elimination полягає в тому, щоб починаючи з повного простору ознак, видаляти по одній найменш значущій (наприклад, з максимальним р-значенням), поки не буде досягнуто бажаної якості моделі або не відбудеться значне її погіршення.

Forward Selection полягає в побудові повного набору однофакторних моделей і вибору найбільш якісної, а також у подальшому додаванні змінних по одній для досягнення бажаної якості моделі. Цей метод значно більш трудомісткий.

Bidirectional Elimination об'єднує два перераховані вище методи відбору змінних.

Найбільш надійний, але й обчислювально найскладніший спосіб заснований на переборі: модель навчається на декількох підмножинах змінних, результати запам'ятовуються, та порівнюються якість моделей. Такий підхід називається Exhaustive Feature Selection.

**Перехресна перевірка (крос-валідація).** Для оцінювання достовірності моделі з метою перевірки, чи узагальнюються результати моделювання на незалежному наборі даних, використовується перехресна перевірка (крос-валідація, англ. cross-validation).

Одноразова перехресна перевірка передбачає розбиття вибірки на взаємодоповнювані під вибірки з метою проведення аналізу на одній частині (що називається навчальним набором, англ. training set) та перевірки результатів на іншій частині (що називається тестовим набором, англ. testing set). Для зниження дисперсії здійснюється багаторазова перехресна перевірка із застосуванням різних розбиттів, результати цих перевірок усереднюють.

Типи крос-валідації:

1. *Крос-валідація на K блоках (K-fold cross-validation).* В цьому випадку набір даних розбивається на  $K$  одинакових за розміром блоків. З  $K$  блоків один залишається для тестування моделі, а інші  $K-1$  використовуються як тренувальний набір. Процес повторюється  $K$  раз, кожен з блоків використовується як тестовий набір один раз. Отримані  $K$  результати усереднюються або комбінуються будь-яким іншим способом, і дають одну оцінку. Перевага такого способу перед випадковим семплюванням (random subsampling) в тому, що всі спостереження використовуються і для тренування, і для тестування моделі, при чому кожне спостереження використовується для тестування лише один раз.

2. *Валідація випадковим семплюванням (random subsampling).* Цей метод випадко розбиває набір даних на тренувальний і тестовий набори. Для кожного такого розбиття модель підлаштовується під тренувальні дані, а

точність прогнозу оцінюється на тестовому наборі. Результати усереднюють за всім розбиттям. Перевага такого методу перед крос-валідацією на  $K$  блоках в тому, що пропорції тренувального та тестового наборів не залежать від кількості блоків. Недолік методу в тому, що деякі спостереження можуть жодного разу не потрапити в тестовий набір, тоді як інші можуть потрапити в нього більше, ніж один раз. Крім того, оскільки розбиття проводяться випадково, результати будуть відрізнятися в разі повторного аналізу.

3. *По-елементна крос-валідація (Leave-one-out, LOO).* В цьому випадку окреме спостереження використовується в якості тестового набору даних, а решта спостережень з вихідного набору – в якості тренувального. Цикл повторюється, поки кожне спостереження не буде використане один раз в якості тестового. Це аналог  $K$ -блокової крос-валідації, де  $K$  дорівнює кількості спостережень у вихідному наборі даних.

## Лабораторна робота 2

### DATA PREPARATION

#### Download data

```
#Set Working Directory
#setwd('D:/ML')
#OR Choose your Directory in 'Files' and click on 'More' -> 'Set as
Working Directory'
#Download file to the table. Source file is 'flats.csv'
f <- read.csv2('flats.csv', header = TRUE, encoding = 'UNICOD')
#Connect Library
library (dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

#Have a Look at the data
glimpse(f)

## Observations: 217
## Variables: 6
## $ rooms      <int> 2, 1, 3, NA, 1, 3, 2, 3, 1, 2, 1, 1, 2, 2, 6, 1, 1,
...
## $ location   <fct> suburbs, center, suburbs, suburbs, suburbs, center,
...
## $ condition  <fct> repaired, repaired, repaired, repaired, repaired,
re...
## $ m2         <int> 50, 37, 67, 21, 82, 82, 45, 82, 41, 63, 41, 38, 45,
...
## $ type       <fct> used, used, used, used, NA, used, used, used, new,
n...
## $ price      <int> 35000, 35000, 65000, 15000, 60000, 85000, 48000,
850...
head(f)

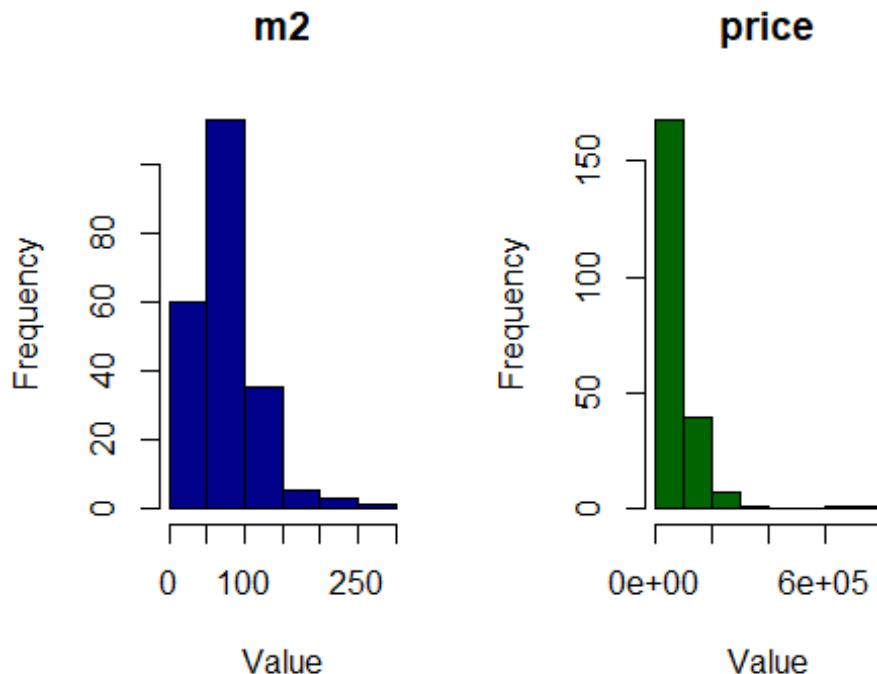
##   rooms location condition m2 type price
## 1      2    suburbs   repaired  50  used 35000
## 2      1     center   repaired  37  used 35000
## 3      3    suburbs   repaired  67  used 65000
## 4     NA    suburbs   repaired  21  used 15000
## 5      1    suburbs   repaired  82 <NA> 60000
## 6      3     center   repaired  82  used 85000
```

Висновок: кількість спостережень – 217, кількість змінних – 6.

## Visualising

### Histogram

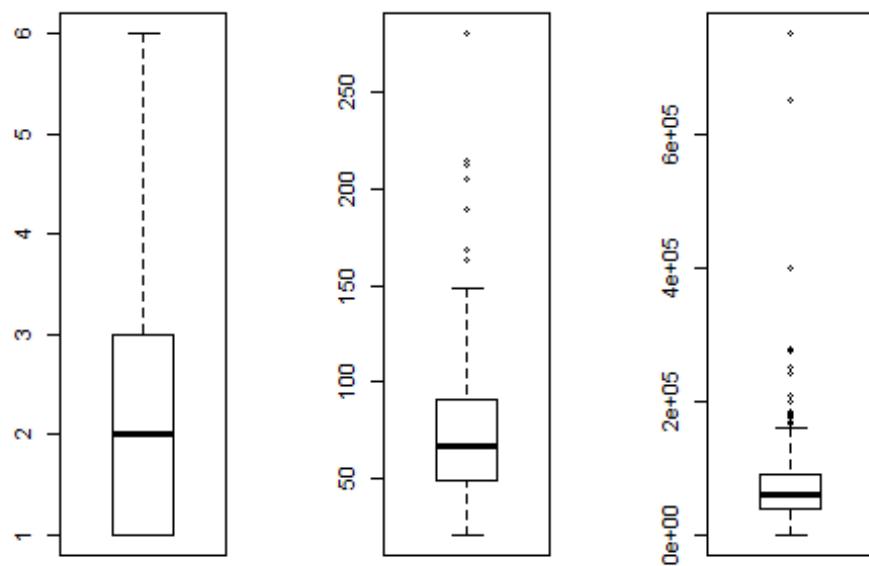
```
library(ggplot2)
par(mfrow = c(1, 2))
hist(f$m2, col = 'dark blue', main = 'm2', xlab = 'Value')
hist(f$price, col = 'dark green', main = 'price', xlab = 'Value')
```



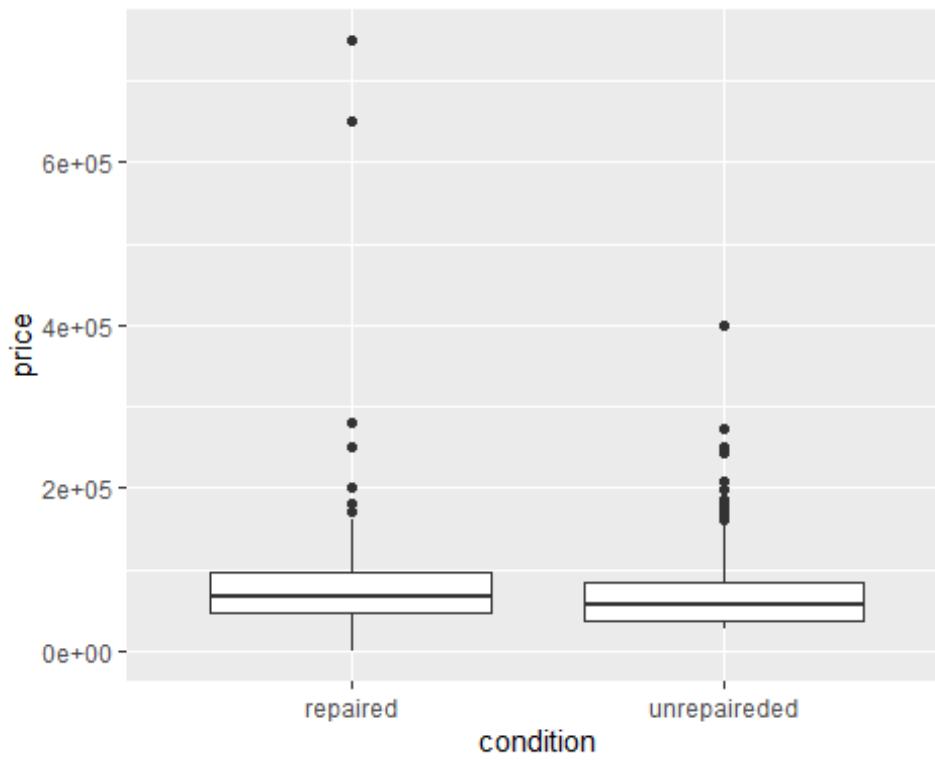
Висновок: розподіл змінної price має довгий хвіст.

### Box-plot

```
par(mfrow = c(1, 3))
boxplot(f$rooms)
boxplot(f$m2)
boxplot(f$price)
```



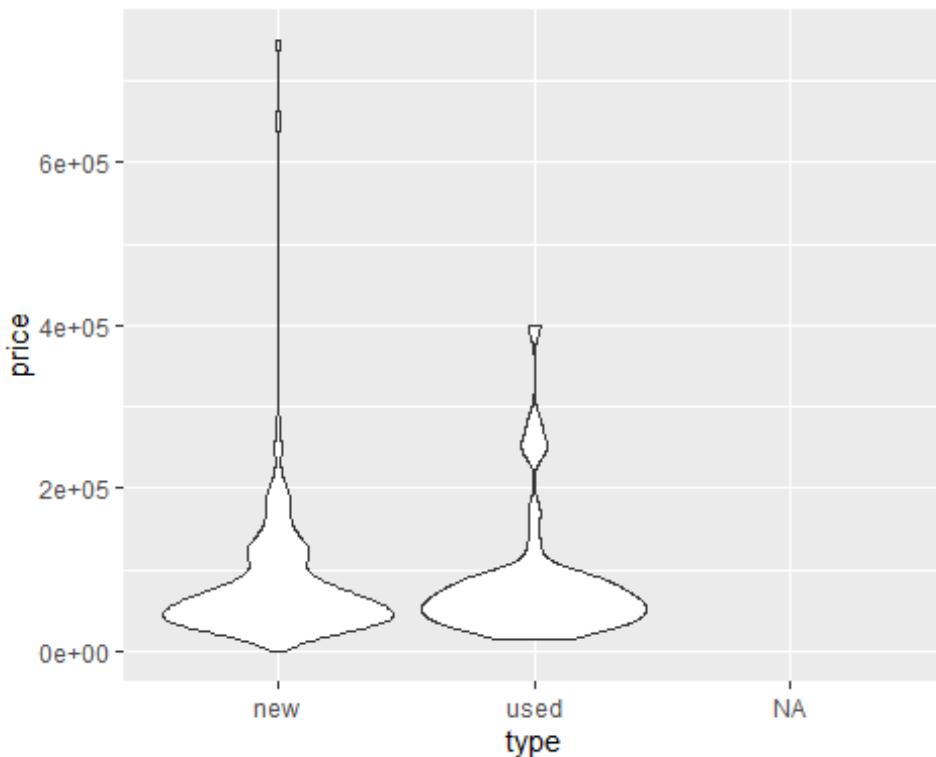
```
qplot(data = f,
      x = condition,
      y = price,
      geom = "boxplot")
```



Висновок: змінні m2 та price мають викиди.

## Violin

```
qplot(data = f,
      x = type,
      y = price,
      geom = "violin")
```



Висновок: більшість нетипових значень змінної price належать до нової нерухомості.

## Statistics

### Descriptive statistics

```
library (psych)

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha

describe(f)

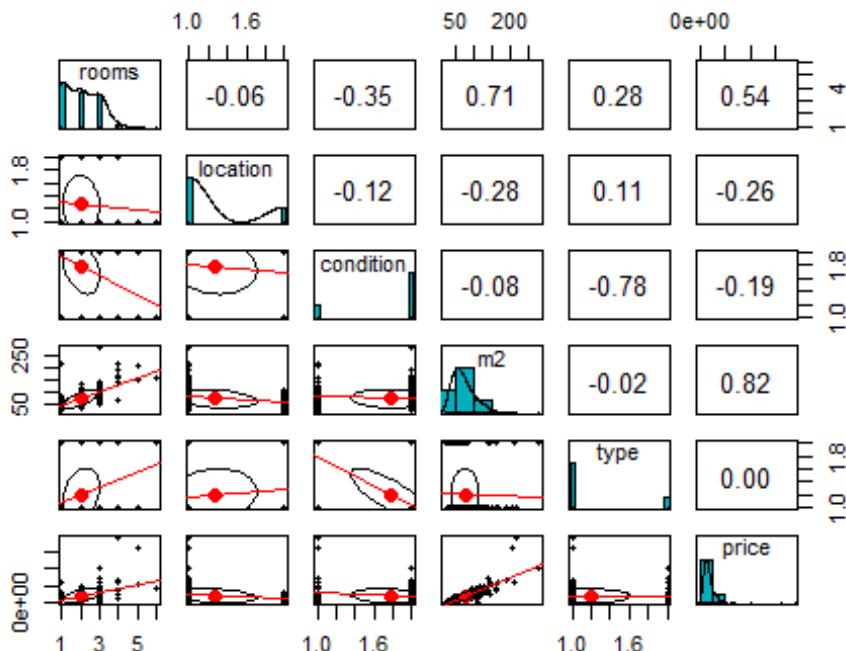
##          vars   n    mean      sd median trimmed      mad min   max
## rooms       1 216    2.01    0.97      2    1.94    1.48    1     6
## location*   2 217    1.27    0.44      1    1.21    0.00    1     2
## condition*  3 217    1.77    0.42      2    1.84    0.00    1     2
## m2          4 217   76.33   38.02     67   70.94   28.17   21   280
```

```
## type*      5 216     1.20     0.40      1     1.13     0.00      1     2
## price      6 217 82427.45 82183.66  59548 67365.84 35609.09      1 750000
## range      skew kurtosis      se
## rooms       5   0.73    0.44    0.07
## location*   1   1.04   -0.91    0.03
## condition*  1  -1.30   -0.30    0.03
## m2          259  1.77    4.61    2.58
## type*       1   1.46    0.14    0.03
## price      749999  4.58   29.38  5578.99
```

**Висновок:** аналіз основних показників описової статистики за кожною змінною показав, що є пропущені значення в змінних – rooms, type. Змінні m2 та price мають викиди.

## Correlations

```
pairs.panels(f, lm=TRUE, # Linear fit
              method = "pearson", # correlation method
              hist.col = "#00AFBB"
              )
```



**Висновок:** найбільше на price впливає змінна m2.

## Missing data

### Delete N/A (not recommended strategy)

```
f_reduce <- tidyverse::drop_na(f)
cat('there are', nrow(f_reduce), 'rows in the f_reduce')

## there are 215 rows in the f_reduce
```

Після видалення пропущених значень в базі залишилося би 215 рядків.

## Fill n/a

```
#1 with neighboring values
f_fill1 <- tidyr::fill(f, rooms, .direction = 'down')
f_fill1 <- tidyr::fill(f_fill1, type, .direction = 'up')
head(f_fill1)

##   rooms location condition m2 type price
## 1      2    suburbs  repaired 50 used 35000
## 2      1     center  repaired 37 used 35000
## 3      3    suburbs  repaired 67 used 65000
## 4      3    suburbs  repaired 21 used 15000
## 5      1    suburbs  repaired 82 used 60000
## 6      3     center  repaired 82 used 85000

#2 with average or the most frequent
f_fill2 <- f
##with average for integer vars
f_fill2$rooms <- ifelse(is.na(f$rooms), round(mean(f$rooms, na.rm = TRUE)), f$rooms)
##the most frequent for categorical vars
f_fill2$type <- ifelse(is.na(f$type), which.max(table(f$type)), f$type)
f_fill2$type <- as.factor(f_fill2$type)
head(f_fill2)

##   rooms location condition m2 type price
## 1      2    suburbs  repaired 50    2 35000
## 2      1     center  repaired 37    2 35000
## 3      3    suburbs  repaired 67    2 65000
## 4      2    suburbs  repaired 21    2 15000
## 5      1    suburbs  repaired 82    1 60000
## 6      3     center  repaired 82    2 85000

#Let's work with "filled with ave" data
f <- f_fill2
```

**Висновок:** для заповнення пропусків обрано варіант заповнення середніми для кількісних змінних і найбільш частотними для якісних змінних.

## Ejections (outside the three sigma)

### Remove the ejections (not recommended strategy)

```
f_ej1 <- f[f$price < mean(f$price)+sd(f$price)*3, ]
describe(f_ej1$price)

##      vars     n      mean       sd median trimmed      mad min     max
range
## X1      1 214 75171.76 52136.84  59042 65832.15 35520.13     1 280000
279999
##      skew kurtosis     se
## X1 1.72      2.92 3564
```

Висновок: після видалення викидів змінної price в базі залишилося би 214 рядків.

### Replace with max

```
f_ej2 <- f
f_ej2$price <- ifelse(f$price <
mean(f$price)+sd(f$price)*3,f$price,mean(f$price)+sd(f$price)*3)
describe(f_ej2$price)

##      vars     n     mean       sd median trimmed      mad min     max
## X1      1 217 78680.61 59689.36 59548 67365.84 35609.09    1 328978.4
##          range skew kurtosis      se
## X1 328977.4 2.02     4.42 4051.98

#let's work with f_eg2
f <- f_ej2
```

Висновок: для корекції викидів обраний варіант заповнення граничними значеннями.

### Splitting the dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f$price, SplitRatio = 0.8)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
#Write prepared data to the file
write.csv2(f_train, file = "flats_train.csv")
write.csv2(f_test, file = "flats_test.csv")
```

Висновок: датасет розподілений на навчальну та тестову вибірки. Результати збережені в окремих файлах.

### **Питання для самоперевірки**

1. Перерахуйте основні завдання інтелектуального аналізу даних.
2. Перерахуйте методи машинного навчання.
3. Що таке CRISP-методологія?
4. Перерахуйте основні етапи CRISP-методології.
5. Перерахуйте етапи підготовки даних.
6. Перерахуйте методи обробки пропущених значень.
7. У чому відмінність кількісного та категоріального кодування?
8. Чим керуватися під час ухвалення рішення про видалення викидів?
9. Як і з якою метою проводиться шкалювання даних?
10. Перерахуйте методи відбору ознак.

### **Самостійна робота 1**

Зібрати дані, що підлягають прогнозуванню (наприклад, характеристики та ціни на будь-які товари). Провести аналіз і підготовку даних.

## РОЗДІЛ 2. РЕГРЕСІЯ

### Тема 4. Лінійна регресія

Лінійна регресія (англ. Linear regression) – модель лінійної залежності однієї (пояснюваної, залежної) змінної у від іншої або кількох інших змінних (факторів, регресорів, незалежних змінних)  $x$ .

У розділі розглянуті три типи моделей:

- 1) однофакторна лінійна регресія:  $Y = a_0 + a_1x + \varepsilon$ ;
- 2) багатофакторна лінійна регресія:  $Y = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n + \varepsilon$ ;
- 3) поліноміальна лінійна регресія:  $Y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n + \varepsilon$ ,

де  $X = \{x_1, x_2, \dots, x_n\}$  – набір вхідних значень,  $y$  – вихід моделі,  $a_i$  – параметри моделі,  $\varepsilon$  – похибка.

У класичній лінійній регресії передбачається, що поряд зі стандартною умовою  $M(\varepsilon) = 0$  виконані також такі припущення:

- гомоскедастичність (постійна або однацова дисперсія) або відсутність гетероскедастичності випадкових помилок моделі  
 $D(\varepsilon) = \sigma^2 = const$ ;
- немає автокореляції випадкових помилок  
 $\forall i, j, i \neq j cov(\varepsilon_i, \varepsilon_j) = 0$ .

Для оцінки якості лінійної моделі використовуються такі характеристики: коефіцієнт детермінації ( $R^2$ ), середньоквадратична похибка ( $MSE$ ), значущість рівняння регресії перевіряється за допомогою  $F$ -критерію Фішера.

Коефіцієнт детермінації ( $R^2$ ) – це частка дисперсії залежної змінної, що пояснюється розглянутою моделлю, тобто пояснівальними змінними

$$R^2 = 1 - \sum_{i=1}^n \frac{(y_i - \hat{y}_i)^2}{(y_i - \bar{y}_i)^2}$$

Середньоквадратичне відхилення або середньоквадратична похибка – міра відмінностей між значеннями, передбаченими моделлю і спостережуваними значеннями

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

*F*-тест або критерій Фішера (*F*-критерій) – статистичний критерій, тестова статистика якого за умови виконання нульової гіпотези має розподіл Фішера (*F*-розподіл). Тест проводиться шляхом порівняння значення статистики з критичним значенням відповідного розподілу Фішера за заданого рівня значущості. Якщо значення цієї статистики більше критичного значення за такого рівня значущості, то нульова гіпотеза відкидається, що означає статистичну значущість регресії. В іншому випадку модель визнається незначущою.

Більш зручний спосіб перевірки гіпотез – за допомогою *p*-значення. *p*(*F*) – ймовірність того, що випадкова величина з цим розподілом Фішера перевищить таке значення статистики. Якщо *p*(*F*) менше за рівень значущості  $\alpha$ , то нульова гіпотеза відкидається (тобто модель визнається значущою).

## Лабораторна робота 3

# LINEAR REGRESSION

### Download the data

```
#Source files are here  
#setwd('D:/ML')  
  
##Features scaling is included in the packages we will work with  
  
#Download the files  
f_train <- read.csv2('flats_train.csv', header = TRUE, encoding =  
  'UNICOD')  
f_train <- f_train[,-1]  
f_test <- read.csv2('flats_test.csv', header = TRUE, encoding = 'UNICOD')  
f_test <- f_test[,-1]
```

Висновок: окрім задані навчальна і тестова вибірки, видалені перші стовпчики з індексами об'єктів до кожної з підвибірок.

### Simple Linear Regression (one factor – m2)

#### Fitting Simple Linear Regression to the Training set

```
model_sr <- lm(price ~ m2, f_train)  
summary(model_sr)  
  
##  
## Call:  
## lm(formula = price ~ m2, data = f_train)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -127623  -15980   -1761    11142   113636  
##  
## Coefficients:  
##                 Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -27333.3     4571.0   -5.98 1.27e-08 ***  
## m2          1383.5      53.3    25.96 < 2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 26950 on 171 degrees of freedom  
## Multiple R-squared:  0.7976, Adjusted R-squared:  0.7964  
## F-statistic: 673.9 on 1 and 171 DF,  p-value: < 2.2e-16
```

**Висновок:** обрана змінна значуча, коефіцієнт детермінації 0,8.

## Predicting

```
p_sr <- predict(model_sr, f_test)

r2_sr <- 1 - sum((f_train$price - predict(model_sr,
f_train))^2) / sum((f_train$price - mean(f_train$price))^2)
R2_sr <- cor(f_train$price, fitted(model_sr))^2 #simplier ex.

train_mse_sr <- sum((f_train$price - predict(model_sr,
f_train))^2) / length(f_train$price)
test_mse_sr <- sum((f_test$price - p_sr)^2) / length(p_sr)

r2_sr
## [1] 0.7976142

R2_sr
## [1] 0.7976142

train_mse_sr
## [1] 718078126

test_mse_sr
## [1] 680604174
```

**Висновок:** вручну розраховані коефіцієнти детермінації. Значення середньоквадратичної похибки на навчальній вибірці – 718 078 126, на тестовій вибірці – 680 604 17, тобто перенавчання немає.

## Visualising

```
library(ggplot2)
ggplot() +
  geom_point(aes(f_train$m2, f_train$price), colour = 'red') +
  geom_point(aes(f_test$m2, f_test$price), colour = 'dark green') +
  geom_line(aes(f_test$m2, p_sr), colour = 'blue') +
  ggtitle('Price vs m2') +
  xlab('m2') +
  ylab('price')
```



**Висновок:** на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Multiple Linear Regression (many factors)

### All factors

```
model_mr <- lm(data = f_train, price ~ .)
summary(model_mr)

##
## Call:
## lm(formula = price ~ ., data = f_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -148721  -13008   -1724    12786   102863 
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)              20596.31   15034.87   1.370  0.172556  
## rooms                  -7710.03    3056.06  -2.523  0.012575 *  
## locationsuburbs         -9449.51    4702.02  -2.010  0.046076 *  
## conditionunrepaireded -26706.80    7301.99  -3.657  0.000341 *** 
## m2                      1462.38     76.41   19.138 < 2e-16 *** 
## type                   -12530.70   7324.65  -1.711  0.088983 .  
## ---                     
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 
##
## Residual standard error: 25660 on 167 degrees of freedom
```

```
## Multiple R-squared:  0.8208, Adjusted R-squared:  0.8154
## F-statistic:  153 on 5 and 167 DF,  p-value: < 2.2e-16
```

**Висновок:** змінна type найменш значуча, коефіцієнт детермінації дорівнює 0,82.

## Optimized model

```
#as p-value, Pr(>|t|) of variable "type" is higher than significance Level
#(5%), Let's exclude this variable from the model
model_opt <- lm(data = f_train, price ~ rooms + location + condition + m2)
summary(model_opt)

##
## Call:
## lm(formula = price ~ rooms + location + condition + m2, data = f_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -139547 -13008 -2124    11980   98940
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             -1741.29    7496.36  -0.232 0.816600    
## rooms                  -8253.23    3056.90  -2.700 0.007646 **  
## locationsuburbs        -9382.04    4728.74  -1.984 0.048879 *   
## conditionunrepaireded -17991.41    5261.11  -3.420 0.000787 *** 
## m2                      1482.58     75.93   19.526 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25810 on 168 degrees of freedom
## Multiple R-squared:  0.8177, Adjusted R-squared:  0.8133
## F-statistic: 188.3 on 4 and 168 DF,  p-value: < 2.2e-16
```

**Висновок:** усі змінні значущі, коефіцієнт детермінації трохи зменшився – 0,81.

## Prediction

```
p_mr <- predict(model_opt, f_test)

train_mse_opt <- sum((f_train$price-predict(model_opt,
f_train))^2)/length(f_train$price)
test_mse_opt <- sum((f_test$price-p_mr)^2)/length(p_mr)

train_mse_opt
## [1] 646925011

test_mse_opt
## [1] 574413579
```

Висновок: значення середньоквадратичної помилки покращилися – на навчальній вибірці – 646 925 011, на тестовій вибірці – 574 413 379, тобто перенавчання немає.

## Visualising

```
ggplot() +  
  geom_point(aes(f_train$m2, f_train$price), colour = 'red') +  
  geom_point(aes(f_test$m2, f_test$price), colour = 'dark green') +  
  geom_line(aes(f_test$m2, p_mr), colour = 'blue') +  
  ggtitle('Price vs m2') +  
  xlab('m2') +  
  ylab('price')
```



Висновок: на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Polynomial Linear Regression (one factor - m2)

### Features extending

```
f_train_poly <- f_train[,c('price', 'm2')]  
f_test_poly <- f_test[,c('price', 'm2')]  
f_train_poly$m22 <- f_train_poly$m2^2  
f_train_poly$m23 <- f_train_poly$m2^3  
f_test_poly$m22 <- f_test_poly$m2^2  
f_test_poly$m23 <- f_test_poly$m2^3
```

**Висновок:** додано змінні  $m2^2$  та  $m2^3$ .

### 3 powers

```
model_pr <- lm(data = f_train_poly, price ~ m22 + m23)
summary(model_pr)

##
## Call:
## lm(formula = price ~ m22 + m23, data = f_train_poly)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -125386 -12653   -3349   10831  108378 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.749e+04 3.650e+03  4.791 3.60e-06 ***
## m22         1.155e+01 8.107e-01 14.249 < 2e-16 ***
## m23        -2.635e-02 3.463e-03 -7.608 1.82e-12 ***
## ---      
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 25980 on 170 degrees of freedom
## Multiple R-squared:  0.8131, Adjusted R-squared:  0.8109 
## F-statistic: 369.7 on 2 and 170 DF,  p-value: < 2.2e-16
```

**Висновок:** змінні  $m2^2$  та  $m2^3$  значущі, коефіцієнт детермінації трохи зменшився – 0,81.

### Predicting

```
p_pr <- predict(model_pr, f_test_poly)

train_mse_poly <- sum((f_train_poly$price - predict(model_pr,
f_train_poly))^2)/length(f_train_poly$price)
test_mse_poly <- sum((f_test_poly$price - p_pr)^2)/length(p_pr)

train_mse_poly
## [1] 663283479

test_mse_poly
## [1] 632354351
```

**Висновок:** значення середньоквадратичної помилки трохи зросли на навчальній вибірці – 659 342 728, на тестовій вибірці – 635 313 813, тобто перенавчання немає.

### Visualising

```
ggplot() +
  geom_point(aes(f_train_poly$m2, f_train_poly$price), colour = 'red') +
  geom_point(aes(f_test_poly$m2, f_test_poly$price), colour = 'dark green') +
  geom_line(aes(f_test_poly$m2, p_pr), colour = 'blue') +
  ggtitle('Price vs m2')
```

```
xlab('m2') +  
ylab('price')
```



Висновок: на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Saving results

```
fit <- data.frame(p_sr, p_mr, p_pr)  
write.csv(fit, file = "flats_fit.csv")
```

Висновок: результати моделювання збережені у файлі.

## Довідка: lm

```
lm(formula, data, subset, weights, na.action,  
method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,  
singular.ok = TRUE, contrasts = NULL, offset, ...)
```

### Arguments

- formula an object of class "[formula](#)" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under ‘Details’.
- data an optional data frame, list or environment (or object coercible by [as.data.frame](#)) to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.

<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>weights</code>	an optional vector of weights to be used in the fitting process. Should be <code>NULL</code> or a numeric vector. If non- <code>NULL</code> , weighted least squares is used with <code>weights</code> <code>weights</code> (that is, minimizing <code>sum(w*e^2)</code> ); otherwise ordinary least squares is used. See also ‘Details’,
<code>na.action</code>	a function which indicates what should happen when the data contain <code>NAs</code> . The default is set by the <code>na.action</code> setting of <code>options</code> , and is <code>na.fail</code> if that is unset. The ‘factory-fresh’ default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.
<code>method</code>	the method to be used; for fitting, currently only <code>method = "qr"</code> is supported; <code>method = "model.frame"</code> returns the model frame (the same as with <code>model = TRUE</code> , see below).
<code>model, x, y, qr</code>	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
<code>singular.ok</code>	logical. If <code>FALSE</code> (the default in S but not in R) a singular fit is an error.

An object of class "lm" is a list containing at least the following components:

<code>coefficients</code>	a named vector of coefficients
<code>residuals</code>	the residuals, that is response minus fitted values.
<code>fitted.values</code>	the fitted mean values.
<code>rank</code>	the numeric rank of the fitted linear model.
<code>weights</code>	(only for weighted fits) the specified weights.
<code>df.residual</code>	the residual degrees of freedom.
<code>call</code>	the matched call.
<code>terms</code>	the <code>terms</code> object used.
<code>contrasts</code>	(only where relevant) the contrasts used.
<code>xlevels</code>	(only where relevant) a record of the levels of the factors used in fitting.
<code>offset</code>	the offset used (missing if none were used).
<code>y</code>	if requested, the response used.
<code>x</code>	if requested, the model matrix used.
<code>model</code>	if requested (the default), the model frame used.
<code>na.action</code>	(where relevant) information returned by <code>model.frame</code> on the special handling of <code>NAs</code> .

### Тема 5. Дерева рішень і випадковий ліс

Дерево рішень (decision tree) як алгоритм машинного навчання – об’єднання логічних правил типу «ЯКЩО ... ТО ...» (if-then) в структуру «дерева», створюючи ієрархічну структуру правил. Дерево рішень складається з вузлів, де проводиться перевірка умови і листя (вузлів рішення), що вказують на клас або його середнє значення.

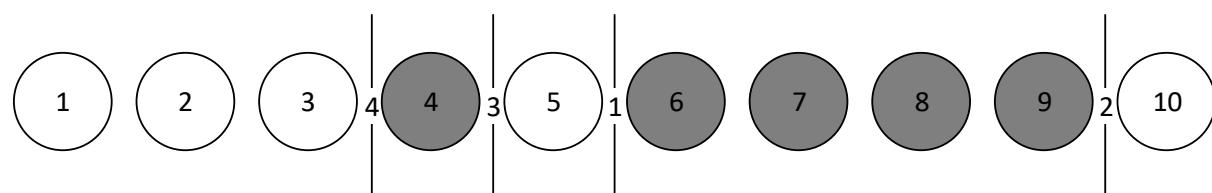
Під час побудови дерева рішень обчислюється приріст інформації (на основі оцінки ентропії). Ентропія відповідає ступеню хаосу в системі. Чим вище ентропія, тим менше впорядкована система і навпаки. Інформація протилежна ентропії. Ентропія Шеннона (Shannon) визначається для системи з  $N$  можливими станами так:

$$H(Y) = -\sum p_i \log_2 p_i$$

де  $p_i$  – ймовірності знаходження системи в  $i$ -му стані.

Основою алгоритмів побудови дерева рішень є принцип жадібної максимізації приросту інформації – на кожному кроці вибирається та ознака, за якою під час розподілу приріст інформації виявляється найбільшим. Далі процедура повторюється рекурсивно, поки ентропія не буде дорівнювати нулю або якісь малій величині (якщо дерево не підлаштовується ідеально під навчальну вибірку, щоб уникнути перенавчання). У різних алгоритмах застосовуються різні евристики для «ранньої зупинки» або «відсікання», щоб уникнути побудови перенавченого дерева.

Розглянемо такий приклад. Нехай є 10 куль, п’ять з яких білі, п’ять – чорні. Вони розміщені послідовно, і потрібно побудувати класифікатор для передбачення кольору кулі.



На початковому етапі ентропія системи максимальна і дорівнює 1.

1. Проведемо перший поділ після номера 5

$$E_{11} = -\left[\frac{1}{5}\log\left(\frac{1}{5}\right) + \frac{4}{5}\log\left(\frac{4}{5}\right)\right] = 0.72, E_{12} = E_{11}.$$

2. Другий поділ проведемо після номера 9

$$E_{21} = 0, E_{22} = E_{21}.$$

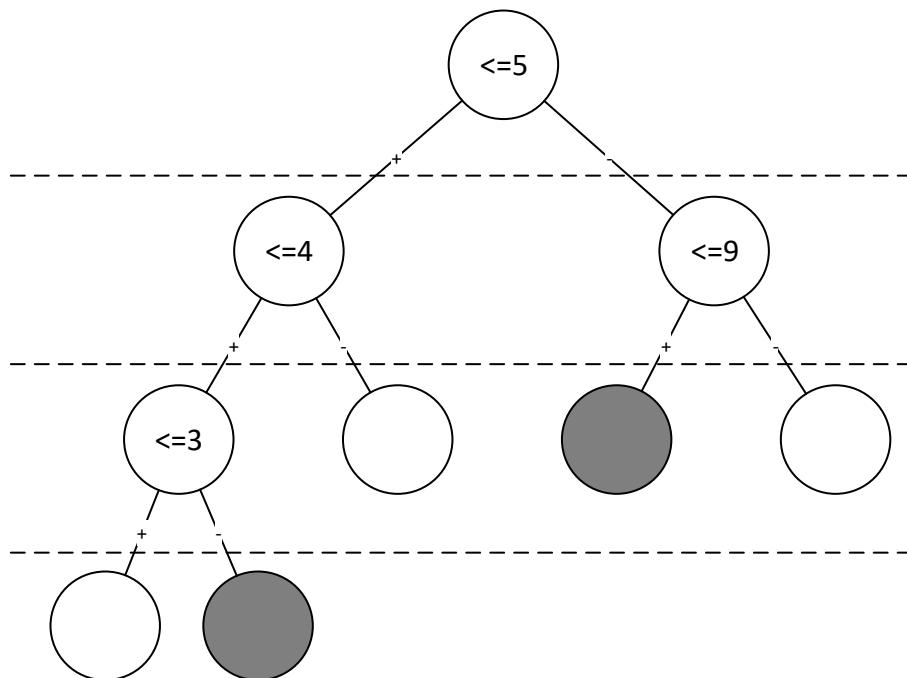
3. Третій поділ проведемо після номера 4

$$E_{31} = -\left[\frac{3}{4}\log\left(\frac{3}{4}\right) + \frac{1}{4}\log\left(\frac{1}{4}\right)\right] = 0.81, E_{32} = 0.$$

4. Четвертий поділ проведемо після номера 3

$$E_{41} = E_{42} = 0.$$

В результаті отримаємо таке дерево рішень



В задачах регресії у вузлах рішень розраховується середнє значення всіх елементів, що потрапили до цього класу. Як прогнозне значення елемента обирається середнє значення типового для нього класу.

### ***Випадковий ліс***

Для підвищення точності моделі з використанням дерев рішень також застосовуються ансамблеві алгоритми машинного навчання, зокрема випадковий ліс, що є одною з реалізацій беггінга.

В межах беггінга на основі вихідної вибірки створюється багато випадкових підвибірок простим вибором з заміщенням. Модель навчається на кожній підвибірці, підсумки роботи усіх моделей усереднюються.

Ефективність беггінга досягається завдяки тому, що базові алгоритми, що пройшли навчання на різних підвибірках, виходять досить різними, їхні помилки взаємно компенсируються, а також за рахунок того, що об'єктивики можуть не потрапляти до деяких навчальних підвибірок. Беггінг ефективний на малих вибірках, коли видалення навіть малої частини навчальних об'єктів призводить до побудови істотно різних моделей.

Випадковий ліс – це беггінг над вирішальними деревами, під час навчання яких для кожного розбиття ознаки обираються з деякої випадкової підмножини ознак.

## Лабораторна робота 4

# NONLINEAR REGRESSION

### Download the data

```
#Source files are here
#setwd('D:/ML')

##Features scaling is included in the packages we will work with

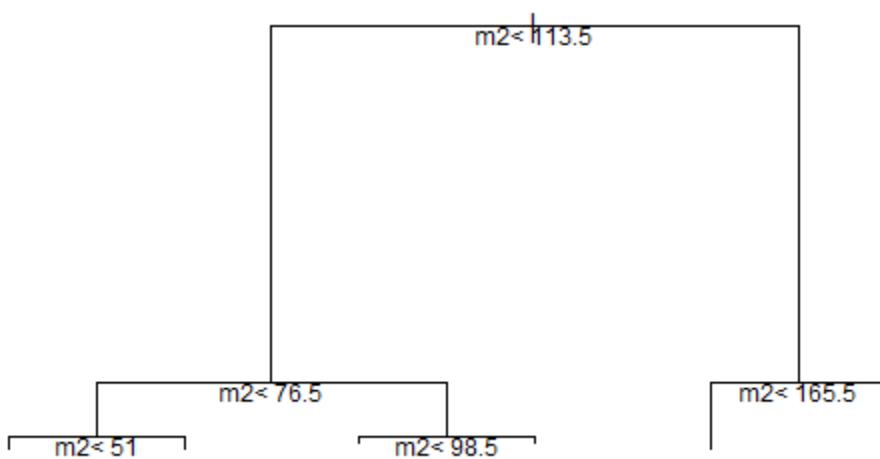
#Download the files
f_train <- read.csv2('flats_train.csv', header = TRUE, encoding =
'UNICOD')
f_test <- read.csv2('flats_test.csv', header = TRUE, encoding = 'UNICOD')
```

**Висновок:** окрім завантажені навчальна і тестова вибірки.

### Decision Tree Regression

#### Fitting

```
# install.packages('rpart')
library(rpart)
model_dt <- rpart(price ~ m2, f_train, control = rpart.control(minsplit =
10))
plot(model_dt)
text(model_dt)
```



**Висновок:** побудовано дерево рішенъ, екзогенна змінна – m2.

#### Predicting

```
p_dt <- predict(model_dt, f_test)
```

```
train_mse_dt <- sum((f_train$price-predict(model_dt,
f_train))^2)/length(f_train$price)
test_mse_dt <- sum((f_test$price-p_dt)^2)/length(p_dt)

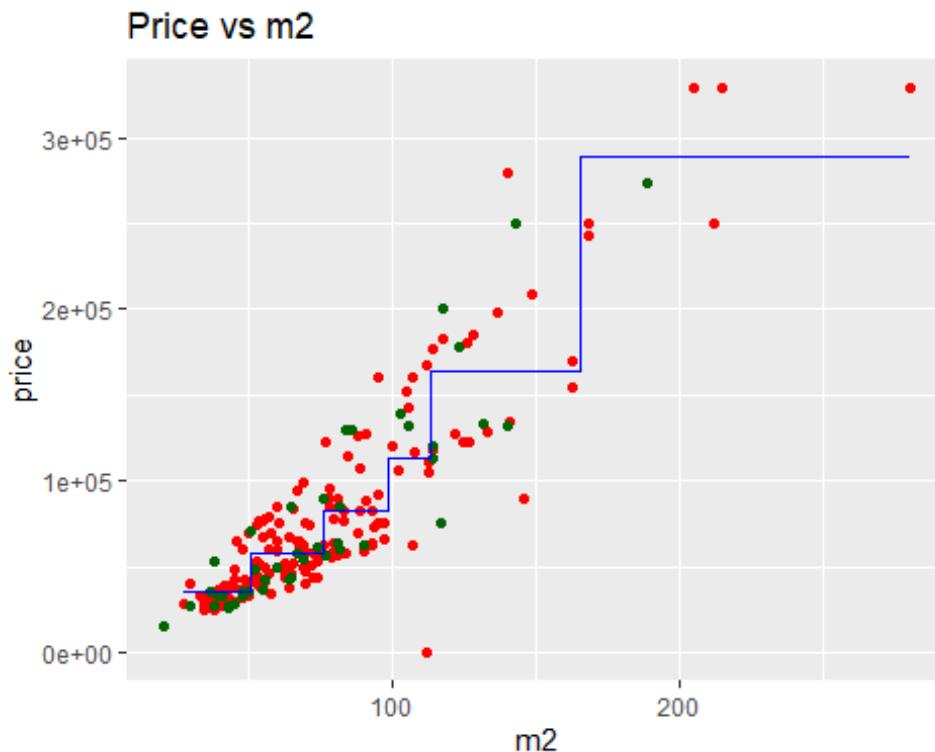
train_mse_dt
## [1] 599249192

test_mse_dt
## [1] 803015457
```

Висновок: значення середньоквадратичної похибки покращилися на навчальній вибірці – 599 249 192, погіршилися на тестовій вибірці – 803 015 457. Модель перенавчено.

## Visualising

```
library(ggplot2)
x_grid <- seq(min(f_train$m2), max(f_train$m2), 0.01)
ggplot() +
  geom_point(aes(f_train$m2, f_train$price), colour = 'red') +
  geom_point(aes(f_test$m2, f_test$price), colour = 'dark green') +
  geom_line(aes(x_grid, predict(model_dt, data.frame(m2 = x_grid))), colour =
  'blue') +
  ggtitle('Price vs m2') +
  xlab('m2') +
  ylab('price')
```



**Висновок:** на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Random forest

### Fitting

```
# install.packages('randomForest')
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

## 
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
## 
##     margin

set.seed(1234)
model_rf = randomForest(x = f_train[m2],
                        y = f_train$price,
                        ntree = 50)
```

**Висновок:** побудовано відліковий ліс із 50 дерев, екзогенна змінна – m2.

### Predicting

```
p_rf <- predict(model_rf, f_test)

train_mse_rf <- sum((f_train$price-predict(model_rf,
f_train))^2)/length(f_train$price)
test_mse_rf <- sum((f_test$price-p_rf)^2)/length(p_rf)

train_mse_rf
## [1] 349901688

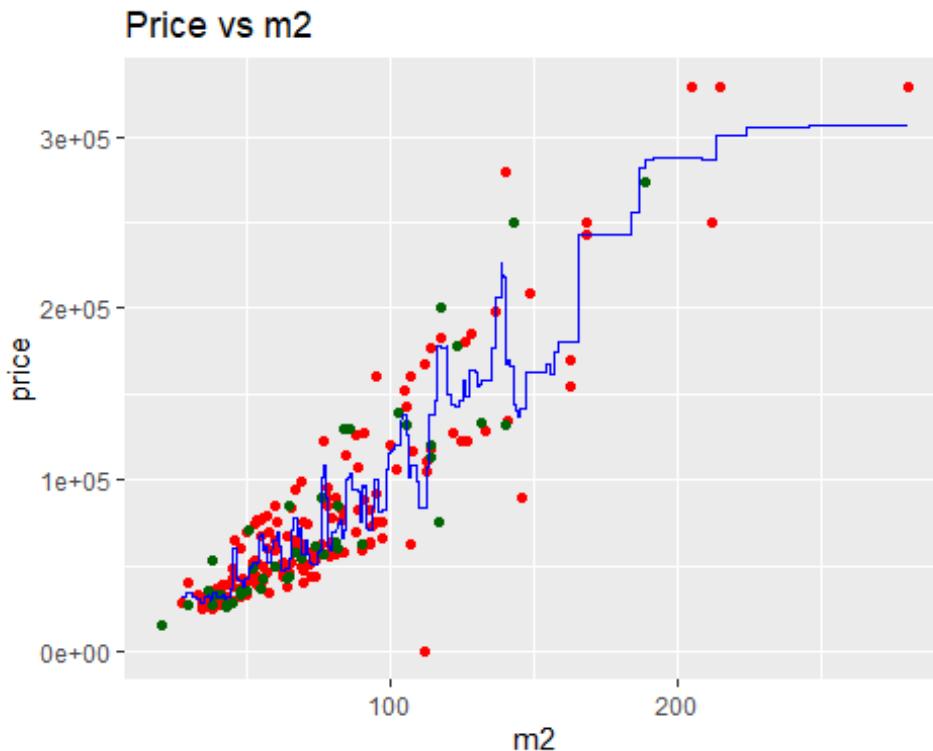
test_mse_rf
## [1] 970138679
```

**Висновок:** значення середньоквадратичної похибки покращилися на навчальній вибірці – 349 901 688, погоршилися на тестовій вибірці – 970 138 679. Модель перенавчено.

### Visualising

```
ggplot() +
  geom_point(aes(f_train$m2, f_train$price), colour = 'red') +
  geom_point(aes(f_test$m2, f_test$price), colour = 'dark green') +
  geom_line(aes(x_grid, predict(model_rf, data.frame(m2 = x_grid))), colour =
'blue') +
  ggtitle('Price vs m2') +
```

```
xlab('m2') +
ylab('price')
```



**Висновок:** на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Saving results

```
fit <- read.csv2('flats_fit.csv', header = TRUE, encoding = 'UNICOD')
fit$p_dt <- p_dt
fit$p_rf <- p_rf
head(fit)

##   X      p_sr      p_mr      p_pr      p_dt      p_rf
## 1 1 23858.084 44860.810 31967.28 35960.83 32465.71
## 2 2 1721.266   3504.312 22337.07 35960.83 32275.88
## 3 3 86117.884 102194.708 80639.08 83140.39 72974.14
## 4 4 86117.884  95070.286 80639.08 83140.39 72974.14
## 5 5 29392.289  32799.704 35090.37 35960.83 31591.74
## 6 6 28008.738  31317.127 34284.27 35960.83 32181.48

write.csv2(fit[-1], file = "flats_fit.csv")
```

**Висновок:** результати моделювання збережені у файлі.

## Довідка: rpart

```
rpart(formula, data, weights, subset, na.action = na.rpart, method,
      model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)
```

## Arguments

<code>formula</code>	a <a href="#">formula</a> , with a response but no interaction terms. If this is a data frame, that is taken as the model frame (see <code>model.frame</code> ).
<code>data</code>	an optional data frame in which to interpret the variables named in the formula.
<code>weights</code>	optional case weights.
<code>subset</code>	optional expression saying that only a subset of the rows of the data should be used in the fit.
<code>na.action</code>	the default action deletes all observations for which <code>y</code> is missing, but keeps those in which one or more predictors are missing.
<code>method</code>	one of "anova", "poisson", "class" or "exp". If <code>method</code> is missing then the routine tries to make an intelligent guess. If <code>y</code> is a survival object, then <code>method = "exp"</code> is assumed, if <code>y</code> has 2 columns then <code>method = "poisson"</code> is assumed, if <code>y</code> is a factor then <code>method = "class"</code> is assumed, otherwise <code>method = "anova"</code> is assumed. It is wisest to specify the method directly, especially as more criteria may be added to the function in future.
	Alternatively, <code>method</code> can be a list of functions named <code>init</code> , <code>split</code> and <code>eval</code> . Examples are given in the file ‘tests/usersplits.R’ in the sources, and in the vignettes ‘User Written Split Functions’.
<code>model</code>	if logical: keep a copy of the model frame in the result? If the input value for <code>model</code> is a model frame (likely from an earlier call to the <code>rpart</code> function), then this frame is used rather than constructing new data.
<code>x</code>	keep a copy of the <code>x</code> matrix in the result.
<code>y</code>	keep a copy of the dependent variable in the result. If missing and <code>model</code> is supplied this defaults to <code>FALSE</code> .
<code>parms</code>	optional parameters for the splitting function. Anova splitting has no parameters. Poisson splitting has a single parameter, the coefficient of variation of the prior distribution on the rates. The default value is 1. Exponential splitting has the same parameter as Poisson. For classification splitting, the list can contain any of: the vector of prior probabilities (component <code>prior</code> ), the loss matrix (component <code>loss</code> ) or the splitting index (component <code>split</code> ). The priors must be positive and sum to 1. The loss matrix must have zeros on the diagonal and positive off-diagonal elements. The splitting index can be <code>gini</code> or <code>information</code> . The default priors are proportional to the data counts, the losses default to 1, and the split defaults to <code>gini</code> .
<code>control</code>	a list of options that control details of the <code>rpart</code> algorithm. See <a href="#">rpart.control</a> .
<code>cost</code>	a vector of non-negative costs, one for each variable in the model. Defaults to one for all variables. These are scalings to be applied when considering splits, so the improvement on splitting on a variable is divided by its cost in deciding which split to choose.
...	arguments to <a href="#">rpart.control</a> may also be specified in the call to <code>rpart</code> . They are checked against the list of valid arguments.

## Порівняння моделей

# Rusults Visualization

### Download the data

```
#Source files are here
#setwd('D:/ML')
#Download the files
f1 <- read.csv2('flats_test.csv', header = TRUE, encoding = 'UNICOD')
f2 <- read.csv2('flats_fit.csv', header = TRUE, encoding = 'UNICOD')
f2 <- f2[-1]
f <- dplyr::bind_cols(f1, f2)
f <- f[,-1]
f$type <- as.factor(f$type)
head(f)

##   rooms location condition m2 type price      p_sr      p_mr      p_pr
## 1     1    center   repaired 37    2 35000 23858.084 44860.810 31967.28
## 2     2   suburbs   repaired 21    2 15000 1721.266 3504.312 22337.07
## 3     1   suburbs   repaired 82    1 60000 86117.884 102194.708 80639.08
## 4     3    center   repaired 82    2 85000 86117.884 95070.286 80639.08
## 5     1  center unrepaireded 41    1 33000 29392.289 32799.704 35090.37
## 6     1  center unrepaireded 40    1 33000 28008.738 31317.127 34284.27
##   p_dt      p_rf
## 1 35960.83 32465.71
## 2 35960.83 32275.88
## 3 83140.39 72974.14
## 4 83140.39 72974.14
## 5 35960.83 31591.74
## 6 35960.83 32181.48
```

### Compare models

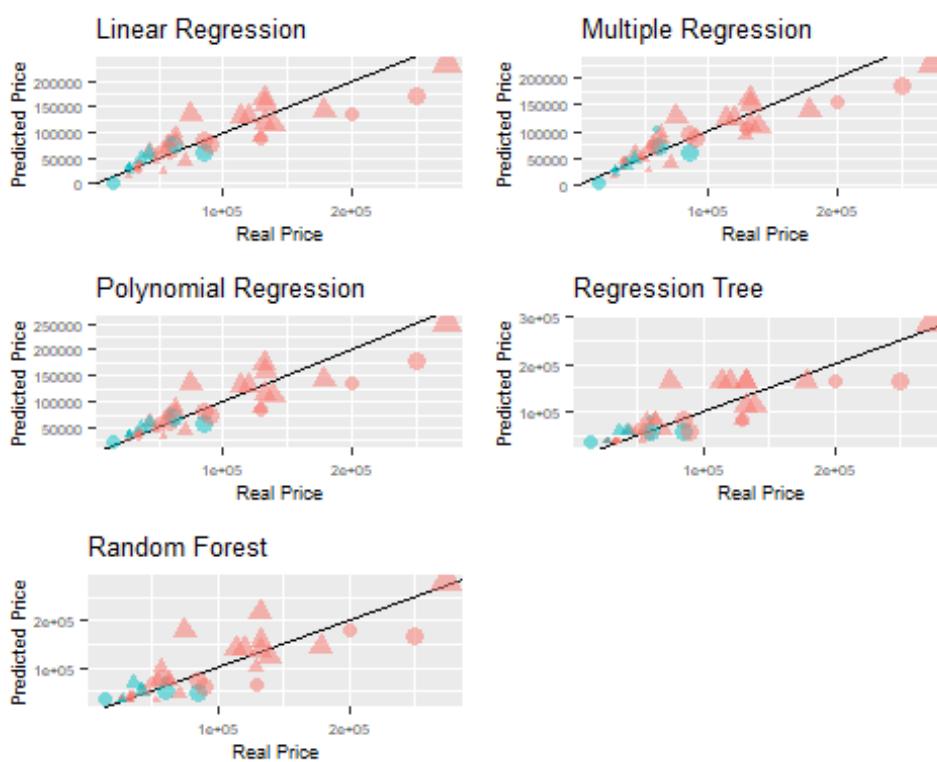
```
g_sr <- ggplot(f, aes(x=price, y=p_sr)) +
  geom_abline(intercept=0, slope=1) +
  geom_point(aes(color=location, shape=condition), size=f$rooms,
alpha=0.5) + labs(title="Linear Regression", x="Real Price", y="Predicted Price") +
  theme(plot.title=element_text(size=10),
axis.title.x=element_text(size=7), axis.title.y=element_text(size=7),
axis.text.x=element_text(size=5), axis.text.y=element_text(size=5)) +
theme(legend.position="none")
g_mr <- ggplot(f, aes(x=price, y=p_mr)) +
  geom_abline(intercept=0, slope=1) +
  geom_point(aes(color=location, shape=condition), size=f$rooms,
alpha=0.5) + labs(title="Multiple Regression", x="Real Price",
y="Predicted Price") +
  theme(plot.title=element_text(size=10),
axis.title.x=element_text(size=7), axis.title.y=element_text(size=7),
axis.text.x=element_text(size=5), axis.text.y=element_text(size=5)) +
theme(legend.position="none")
```

```

g_pr <- ggplot(f, aes(x=price, y=p_pr)) +
  geom_abline(intercept=0, slope=1) +
  geom_point(aes(color=location, shape=condition), size=f$rooms,
  alpha=0.5) + labs(title="Polynomial Regression", x="Real Price",
  y="Predicted Price") +
  theme(plot.title=element_text(size=10),
axis.title.x=element_text(size=7), axis.title.y=element_text(size=7),
axis.text.x=element_text(size=5), axis.text.y=element_text(size=5)) +
theme(legend.position="none")
g_dt <- ggplot(f, aes(x=price, y=p_dt)) +
  geom_abline(intercept=0, slope=1) +
  geom_point(aes(color=location, shape=condition), size=f$rooms,
  alpha=0.5) + labs(title="Regression Tree", x="Real Price", y="Predicted
Price") +
  theme(plot.title=element_text(size=10),
axis.title.x=element_text(size=7), axis.title.y=element_text(size=7),
axis.text.x=element_text(size=5), axis.text.y=element_text(size=5)) +
theme(legend.position="none")
g_rf <- ggplot(f, aes(x=price, y=p_rf)) +
  geom_abline(intercept=0, slope=1) +
  geom_point(aes(color=location, shape=condition), size=f$rooms,
  alpha=0.5) + labs(title="Random Forest", x="Real Price", y="Predicted
Price") +
  theme(plot.title=element_text(size=10),
axis.title.x=element_text(size=7), axis.title.y=element_text(size=7),
axis.text.x=element_text(size=5), axis.text.y=element_text(size=5)) +
theme(legend.position="none")

gridExtra::grid.arrange(g_sr,g_mr,g_pr,g_dt,g_rf,ncol=2)

```



### Calc prediction error and visualize it

```
sr <- mean ((f$price - f$p_sr) ^ 2)
mr <- mean ((f$price - f$p_mr) ^ 2)
pr <- mean ((f$price - f$p_pr) ^ 2)
dt <- mean ((f$price - f$p_dt) ^ 2)
rf <- mean ((f$price - f$p_rf) ^ 2)
mse <- data.frame(sr,mr,pr,dt,rf)
head(mse)

##           sr          mr          pr          dt          rf
## 1 680604174 574413579 632354351 803015457 970138679

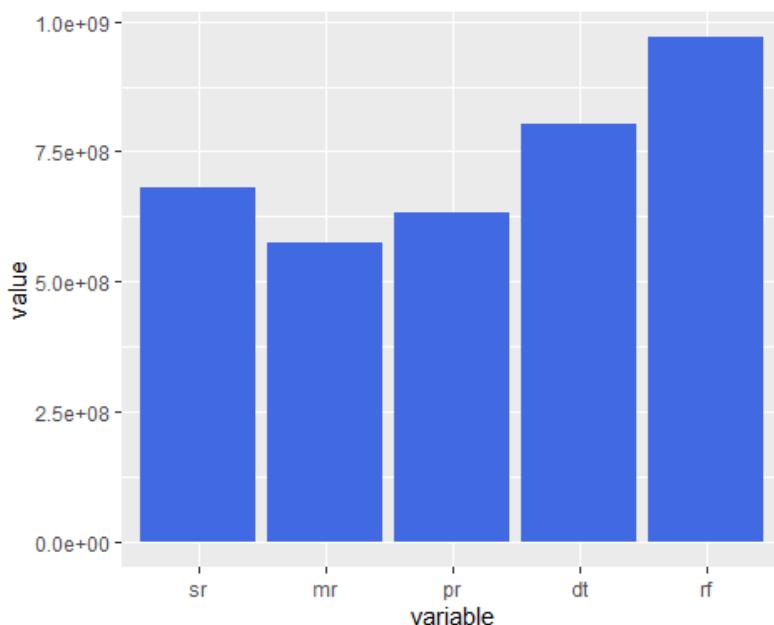
mse1 <- reshape::melt.data.frame(mse)

## Using   as id variables

head(mse1)

##   variable     value
## 1      sr 680604174
## 2      mr 574413579
## 3      pr 632354351
## 4      dt 803015457
## 5      rf 970138679

b1 <- ggplot(mse1, aes(x=variable, y=value)) +
  geom_bar(stat="summary", fun.y="mean", fill = 'royalblue')
b1
```



### Save results

```
ggsave("plot.jpg", plot=b1 + theme_classic(), width=20, height=15,
units="cm", dpi=600)
```

### Переваги і недоліки розглянутих моделей

Модель	Переваги	Недоліки
Лінійна регресія	Працює на вибірках будь-якого обсягу, дозволяє оцінити значущість факторів	Припущення лінійності
Поліноміальна регресія	Працює на вибірках будь-якого обсягу, враховує нелінійність	Необхідність вручну підбирати ступінь полінома
Дерева рішень	Добре інтерпретуються, не потребують шкалювання, дозволяють моделювати лінійні та нелінійні залежності	Погано працюють на малих вибірках, велика ймовірність перенавчання
Випадковий ліс	Висока точність, особливо у разі моделювання нелінійних залежностей	Не інтерпретуються, легко перенавчаються, потрібно вручну підбирати кількість дерев

### Питання для самоперевірки

1. Перелічіть види моделей регресії.
2. Охарактеризуйте типи моделей лінійної регресії.
3. Допущення моделей лінійної регресії.
4. Переваги та недоліки моделей лінійної регресії.
5. Переваги та недоліки моделей поліноміальної регресії.
6. Особливості використання дерев рішення в моделях регресії.
7. Поняття ентропії та його використання під час побудови дерева рішень.
8. Переваги і недоліки дерева рішень в задачах регресії.
9. Особливості моделі випадкового лісу.
10. Переваги і недоліки моделі випадкового лісу в задачах регресії.

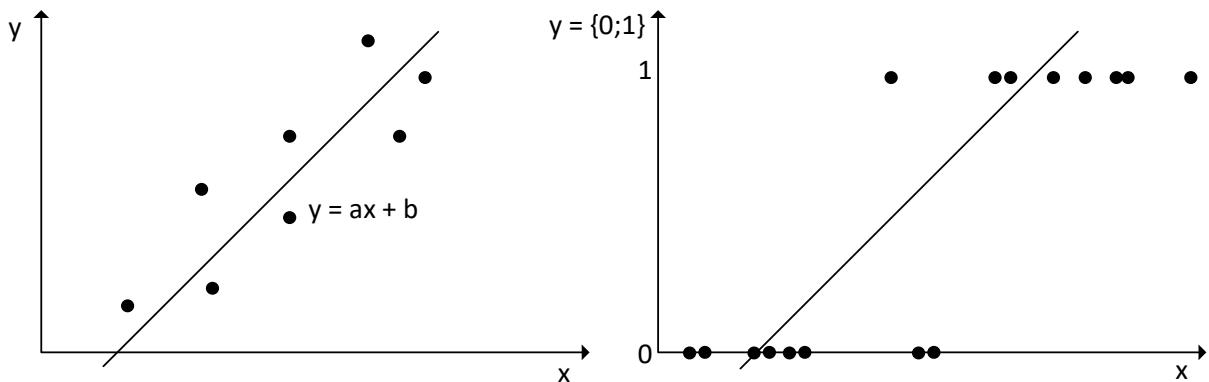
### Самостійна робота 2

На основі зібраних даних побудувати моделі регресії, провести їх порівняльний аналіз і зробити висновки.

### РОЗДІЛ 3. КЛАСИФІКАЦІЯ

#### Тема 6. Логістична регресія

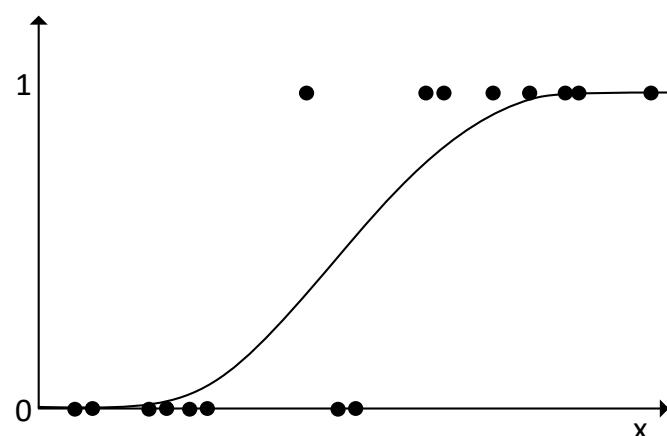
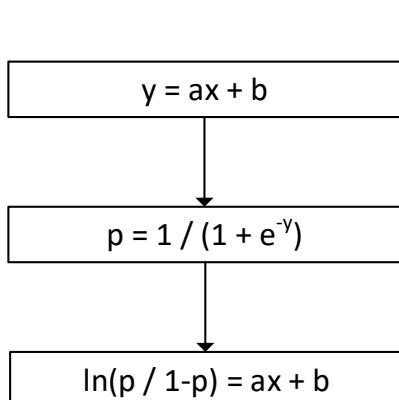
У разі побудови регресійної моделі ендогенна змінна кількісна. У моделях класифікації пояснюється якісна змінна. Розглянемо, як ідея регресії може бути застосована в цьому разі.



Логістична регресія або логіт-регресія (англ. logit model) – це статистична модель, що використовується для передбачення ймовірності виникнення деякої події  $p$  за значеннями множини ознак  $X$

$$p = 1/(1 + e^{-y}),$$

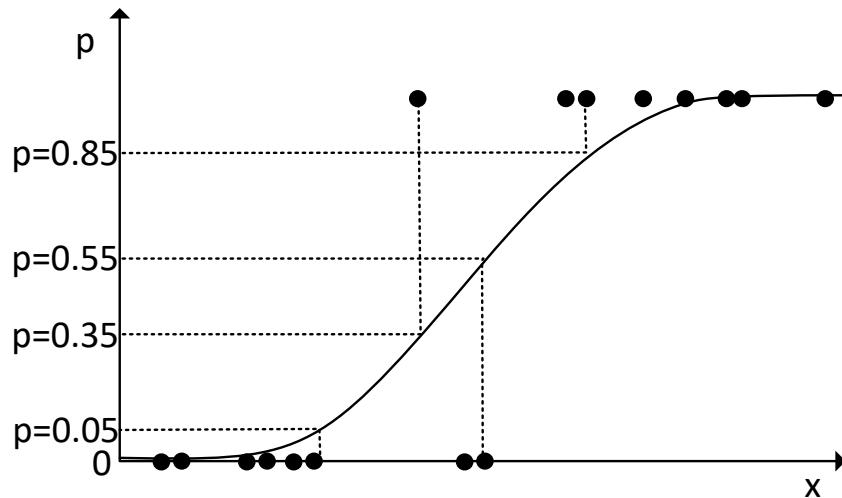
де  $y$  – значення регресії:  $y = a_1x_1 + a_2x_2 + \dots + a_nx_n$ .



Фактично, під час побудови логістичної регресії оцінюється співвідношення шансів OR (odds ratio), тобто співвідношення ймовірності того, що подія відбудеться і ймовірності того, що подія не відбудеться

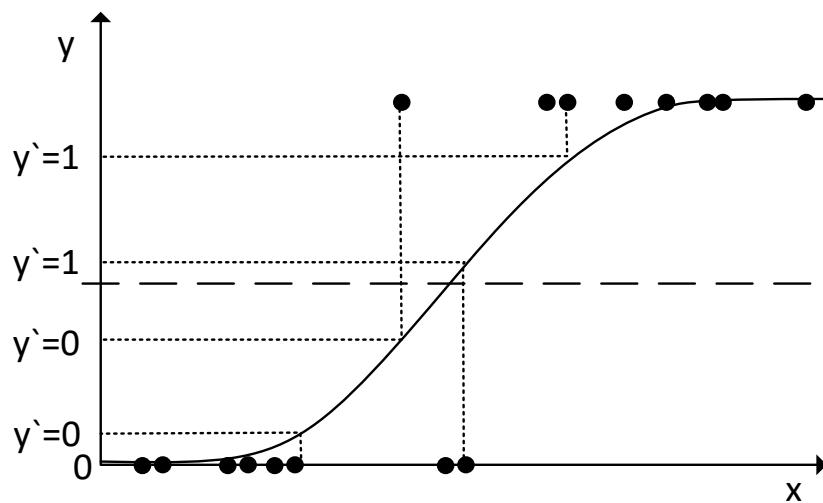
$$OR = p / (1 - p),$$

де  $p$  – ймовірність успіху,  $\log(OR) = y$ .



В межах моделі об'єкт належить до одного з класів  $\{0; 1\}$  з огляду на те, чи перевищує його оцінка ймовірності поріг відсікання  $k$

$$\hat{y} = \{0, p < k; 1, p \geq k\}$$

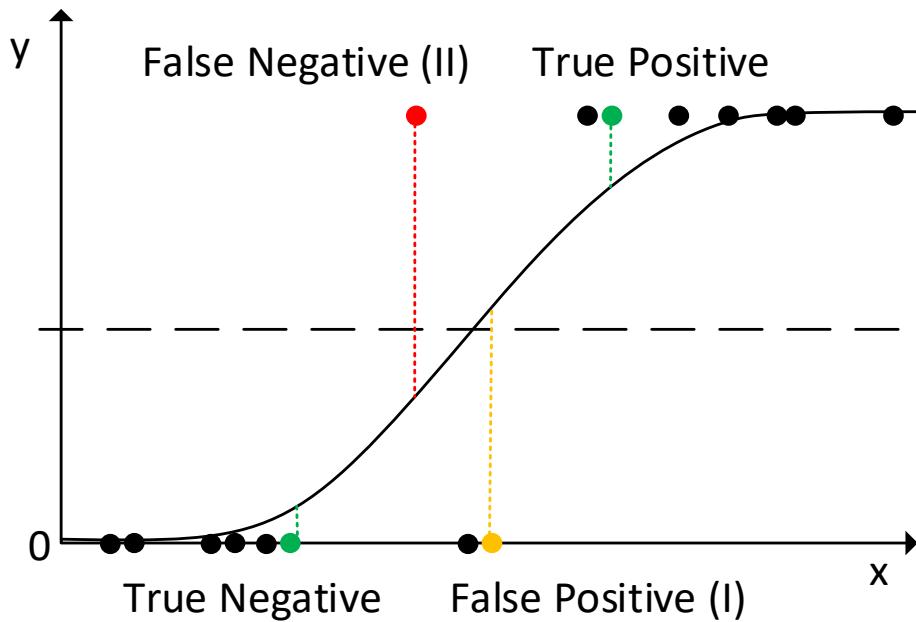


Для оцінки коефіцієнтів логістичної регресії використовують метод максимальної правдоподібності. Основою методу є функція правдоподібності (likelihood function), що виражає щільність ймовірності спільної появи результатів вибірки.

Для побудови логістичної регресії навчальна вибірка готується стандартно з тією відмінністю, що вихідне поле може бути тільки дискретного типу. На етапі визначення входів моделі необхідно пам'ятати, що для успішного навчання кількість прикладів має в кілька разів перевищувати кількість вхідних ознак. За малої кількості даних доводиться штучно спрошувати структуру регресійній моделі, залишаючи найбільш істотні ознаки. Для вихідного поля (залежної змінної) необхідно визначити, що є негативною (negative), а що позитивною подією (positive). Це залежить від конкретного завдання. *Наприклад, якщо прогнозується ймовірність наявності захворювання, то позитивним результатом буде клас «Хворий пацієнт», негативним – «Здоровий пацієнт».*

В результаті побудови моделі виникають чотири варіанти класифікації:

- $TP$  (True Positives) – правильно класифіковані позитивні приклади (істинно позитивні випадки);
- $TN$  (True Negatives) – правильно класифіковані негативні приклади (істинно негативні випадки);
- $FN$  (False Negatives) – позитивні приклади, класифіковані як негативні (помилка II типу). Це так званий «помилковий пропуск», коли подія, що цікавить нас помилково не виявляється (хибнонегативні приклади);
- $FP$  (False Positives) – негативні приклади, класифіковані як позитивні (помилка I типу). Це помилкове виявлення, тому що за браком подій помилково ухвалюється рішення про її наявність (хибнопозитивні випадки).



Для оцінки якості моделі використовується таблиця спряженості, що будується на основі результатів класифікації.

		Модель	
		Негативно	Позитивно
Фактично	Негативно	TN	FP (I)
	Позитивно	FN (II)	TP

На основі таблиці спряженості оцінюються:

- точність моделі

$$\text{Accuracy Rate} = \frac{TP + TN}{Total}$$

- частка помилок

$$\text{Error Rate} = \frac{FP + FN}{Total}$$

- чутливість (Sensitivity) – частка істинно позитивних випадків, що були правильно ідентифіковані моделлю

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

- специфічність (Specificity) – частка істинно негативних випадків, які були правильно ідентифіковані моделлю

$$Sp = \frac{TN}{TN + FP}$$

Модель з високою чутливістю часто дає істинний результат при наявності позитивного результату (виявляє позитивні приклади). Навпаки, модель з високою специфічністю частіше дає істинний результат за наявності негативного результату (виявляє негативні приклади).

		Predicted	
		False	True
Actual	No	35	5
	Yes	10	50

$$\text{Accuracy Rate} = \text{Correct} / \text{Total}$$

$$AR = 85 / 100 = 85\%$$

$$\text{Error Rate} = \text{Wrong} / \text{Total}$$

$$ER = 15 / 100 = 15\%$$

Також для оцінки моделі використовується ROC-аналіз з використанням графіків – ROC-кривих (Receiver Operator Characteristic). Оскільки класів два, один з них називається класом з позитивними наслідками, другий – з негативними. ROC-крива показує залежність кількості правильно класифікованих позитивних прикладів від кількості неправильно класифікованих негативних прикладів. У термінології ROC-аналізу перші називаються істинно позитивною, другі – хибно негативною множиною.

ROC-крива будується так: для кожного значення порога відсікання<sup>2</sup> розраховуються значення чутливості  $Se$  і специфічності  $Sp$ . Будується графік залежності: по осі  $Y$  відкладається чутливість  $Se$ , по осі  $X$  –  $(100 \% - Sp)$ .

Для ідеального класифікатора графік ROC-кривої проходить через верхній лівий кут, де частка істинно позитивних випадків становить 100 % або 1,0 (ідеальна чутливість), чим ближче крива до верхнього лівого кута, тим вище здатність до передбачування моделі.

Ідеальній моделі притаманна 100 % чутливість та специфічність. Однак на практиці досягти цього неможливо, більш того, неможливо одночасно підвищити і чутливість, і специфічність моделі. Компроміс знаходиться за допомогою порога відсікання, тому що порогове значення впливає на співвідношення  $Se$  і  $Sp$ .

---

<sup>2</sup> Починаючи з якого рівня відносити приклад до того чи іншого класу. Поріг відсікання змінюється від 0 до 1 з кроком (наприклад, 0,01).

## Лабораторна робота 5

# LOGISTIC REGRESSION

### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
```

Висновок: для побудови моделі банківського скорингу використані дані про наявність прострочених платежів по кредиту.

### Statistics

#### Descriptive statistics

```
library (psych)
describe(f)

##          vars   n     mean      sd median trimmed    mad    min
## id*       1 600 300.50 173.35 300.5 300.50 222.39 1.00
## age       2 600 42.40 14.42  42.0  42.34 17.79 18.00
## sex*      3 600 1.50  0.50   1.5   1.50  0.74 1.00
## region*   4 600 2.23  1.29   2.0   2.16  1.48 1.00
## income    5 600 27524.03 12899.47 24925.3 26432.88 12854.96 5014.21
## married*  6 600 1.66  0.47   2.0   1.70  0.00 1.00
## children  7 600 1.01  1.06   1.0   0.89  1.48 0.00
## car*      8 600 1.49  0.50   1.0   1.49  0.00 1.00
## mortgage* 9 600 1.35  0.48   1.0   1.31  0.00 1.00
## delays*   10 600 1.51  0.50   2.0   1.51  0.00 1.00
##                  max    range skew kurtosis      se
## id*        600.0 599.00  0.00 -1.21  7.08
## age         67.0  49.00  0.04 -1.17  0.59
## sex*        2.0   1.00  0.00 -2.00  0.02
## region*     4.0   3.00  0.38 -1.58  0.05
## income      63130.1 58115.89 0.66 -0.33 526.62
## married*    2.0   1.00 -0.67 -1.55  0.02
## children    3.0   3.00  0.55 -1.04  0.04
## car*        2.0   1.00  0.03 -2.00  0.02
## mortgage*   2.0   1.00  0.64 -1.60  0.02
## delays*     2.0   1.00 -0.03 -2.00  0.02
```

Висновок: кількість спостережень – 600, кількість змінних – 10, з них якісних – 7, кількісних – 3. Пропущених значень і викидів немає.

### Features Scaling

```
f <- f[-1] #exclude ID column
sc <- f[,c('age','income','children')]
```

```

sc <- scale(sc)
f$age <- sc[,c('age')]
f$income <- sc[,c('income')]
f$children <- sc[,c('children')]
head (f)

##      age    sex   region    income married   children car mortgage
## 1  0.3885629 FEMALE INNER_CITY -0.7735227   NO -0.01104012  NO     NO
## 2 -0.1660318   MALE     TOWN  0.1985406   YES  1.88155126 YES     YES
## 3  0.5965360 FEMALE INNER_CITY -0.8487661   YES -0.95733581 YES     NO
## 4 -1.3445456 FEMALE     TOWN -0.5541803   YES  1.88155126  NO     NO
## 5  1.0124820 FEMALE    RURAL  1.7870712   YES -0.95733581  NO     NO
## 6  1.0124820 FEMALE     TOWN  0.8020151   YES  0.93525557  NO     NO

## delays
## 1  YES
## 2  NO
## 3  YES
## 4  YES
## 5  NO
## 6  NO

```

**Висновок:** моделі класифікації вимагають попереднього шкалювання кількісних змінних.

### Splitting the scaled dataset into the TRAIN set and TEST set

```

set.seed(123)
library(caTools)
split = sample.split(f$delays, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)

```

**Висновок:** підготований датасет розділено на навчальну та тестову вибірки.

### Fitting (Benchmark model)

```

class_lr <- glm(delays ~ ., f_train, family = binomial)
summary(class_lr)

##
## Call:
## glm(formula = delays ~ ., family = binomial, data = f_train)
##
## Deviance Residuals:
##      Min        1Q        Median         3Q        Max 
## -2.45478  -0.27305   0.05274   0.35816   2.48678 
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) 0.1437900  0.4672298  0.308  0.75827    
## age          -0.6301027  0.2380023 -2.647  0.00811 **  
## sexMALE      0.0753371  0.3482577  0.216  0.82873    
## regionRURAL -0.5058248  0.5021533 -1.007  0.31379    
## regionSUBURBAN 0.0005795  0.6181679  0.001  0.99925    

```

```
## regionTOWN    -0.3787819  0.3988669  -0.950  0.34229
## income        -3.6434752  0.4395838  -8.288  < 2e-16 ***
## marriedYES   -0.7017291  0.3861016  -1.817  0.06914 .
## children      0.0130407  0.1712227   0.076  0.93929
## carYES        0.0792534  0.3470280   0.228  0.81935
## mortgageYES  0.1388293  0.3558472   0.390  0.69644
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 554.43 on 399 degrees of freedom
## Residual deviance: 223.61 on 389 degrees of freedom
## AIC: 245.61
##
## Number of Fisher Scoring iterations: 7
```

Висновок: значущими змінними є age та income.

## Optimized model

```
class_opt <- glm(delays ~ age + income, f_train, family = binomial)
summary(class_opt)

##
## Call:
## glm(formula = delays ~ age + income, family = binomial, data = f_train)
##
## Deviance Residuals:
##       Min        1Q     Median        3Q       Max
## -2.13034  -0.26462   0.07112   0.36977   2.33213
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.4259    0.1836 -2.320  0.02035 *
## age         -0.6270    0.2319 -2.704  0.00685 **
## income      -3.5638    0.4209 -8.466  < 2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 554.43 on 399 degrees of freedom
## Residual deviance: 228.42 on 397 degrees of freedom
## AIC: 234.42
##
## Number of Fisher Scoring iterations: 6
```

Висновок: всі змінні оптимізованої моделі є значущими.

## Predicting

```
p <- predict(class_opt, f_test[, c('age', 'income')], type = 'response')
y <- ifelse(p > 0.5, 1, 0)
```

**Висновок:** розраховані ймовірності віднесення об'єктів до кожного з двох класів (вектор  $\hat{y}$ ), визначені класи об'єктів (вектор  $y$ ).

## Confusion Matrix

```
cm = table(f_test[, 'delays'], y > 0.5)
print(cm)

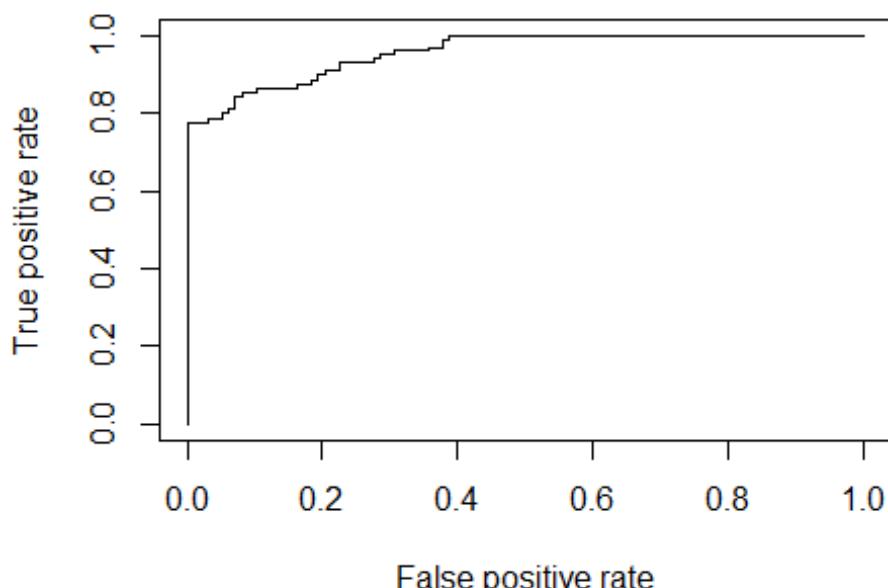
##
##          FALSE  TRUE
##    NO      85    13
##    YES     14    88
```

**Висновок:** точність моделі -  $(85 + 88) / 200 = 86,5\%$ , частка невірно класифікованих випадків -  $(13 + 14) / 200 = 13,5\%$ . Чутливість моделі -  $88 / (14 + 88) = 86\%$ , специфічність -  $85 / (85 + 13) = 87\%$ , тобто модель більш чутлива до виявлення негативних випадків. У цьому разі - кредиторів, що не мають заборгованості.

## ROC

```
library(ROCR)

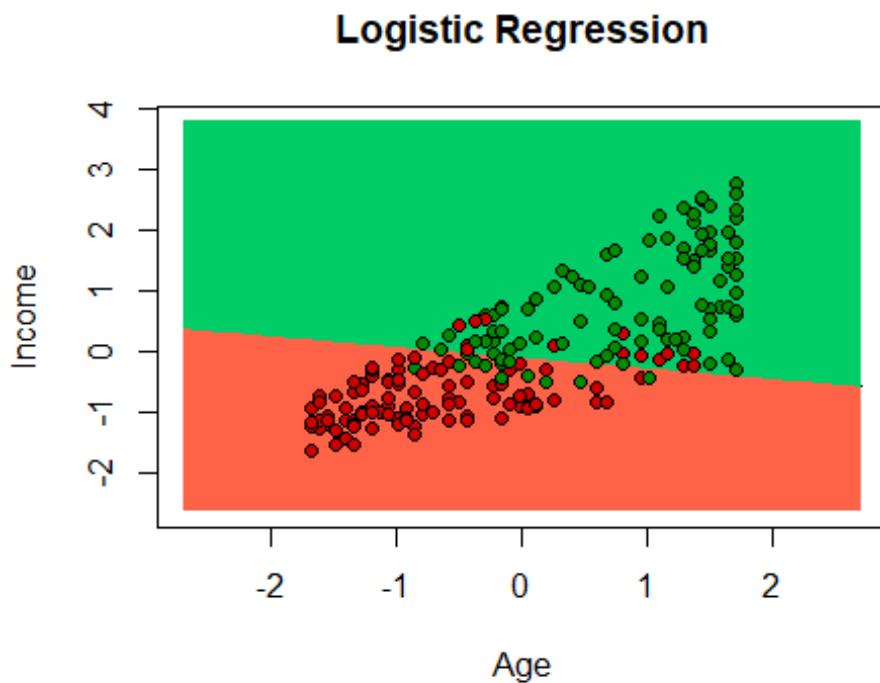
pref <- prediction(p, f_test$delays)
perf <- performance(pref, "tpr", "fpr")
plot(perf)
```



**Висновок:** співвідношення істинно-позитивних і хибно-позитивних випадків свідчить про відносно хорошу якість моделі.

## Visualising the Test set results

```
#install.packages("ElemStatLearn")
library(ElemStatLearn)
set = f_test[,c('age','income','delays')]
X1 = seq(min(set['age']) - 1, max(set['age']) + 1, by = 0.01)
X2 = seq(min(set['income']) - 1, max(set['income']) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('age', 'income')
prob_set = predict(class_opt, grid_set, type = 'response')
y_grid = ifelse(prob_set > 0.5, 1, 0)
plot(set[, -3],
      main = 'Logistic Regression',
      xlab = 'Age', ylab = 'Income',
      xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'tomato',
'springgreen3'))
points(set, pch = 21, bg = ifelse(set[, 3] == 'YES', 'red3', 'green4'))
```



**Висновок:** на графіку червоним позначені випадки затримки повернення кредиту, зеленим – хороши кредитори. Червоним виділена зона високої ймовірності неповернення кредиту. Модель описує лінійний варіант розділяючої кривої.

## Довідка: glm

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, singular.ok = TRUE, contrasts = NULL, ...)

glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE, singular.ok = TRUE)

## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

### Arguments

<code>formula</code>	an object of class " <a href="#">formula</a> " (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under ‘Details’.
<code>family</code>	a description of the error distribution and link function to be used in the model. For <code>glm</code> this can be a character string naming a family function, a family function or the result of a call to a family function. For <code>glm.fit</code> only the third option is supported. (See <a href="#">family</a> for details of family functions.)
<code>data</code>	an optional data frame, list or environment (or object coercible by <a href="#">as.data.frame</a> ) to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>glm</code> is called.
<code>weights</code>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be <code>NULL</code> or a numeric vector.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>na.action</code>	a function which indicates what should happen when the data contain <code>NAs</code> . The default is set by the <code>na.action</code> setting of <a href="#">options</a> , and is <code>na.fail</code> if that is unset. The ‘factory-fresh’ default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.
<code>start</code>	starting values for the parameters in the linear predictor.
<code>etastart</code>	starting values for the linear predictor.
<code>mustart</code>	starting values for the vector of means.
<code>offset</code>	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. This should be <code>NULL</code> or a numeric vector of length equal to the number of cases. One or more <code>offset</code> terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See <a href="#">model.offset</a> .
<code>control</code>	a list of parameters for controlling the fitting process. For <code>glm.fit</code> this is passed to <code>glm.control</code> .
<code>model</code>	a logical value indicating whether <i>model frame</i> should be included as a component of the returned value.

method	the method to be used in fitting the model. The default method " <code>glm.fit</code> " uses iteratively reweighted least squares (IWLS): the alternative " <code>model.frame</code> " returns the model frame and does no fitting.
	User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as <code>glm.fit</code> . If specified as a character string it is looked up from within the stats namespace.
x, y	For <code>glm</code> : logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.
	For <code>glm.fit</code> : x is a design matrix of dimension n * p, and y is a vector of observations of length n.
singular.ok	logical; if FALSE a singular fit is an error.
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
intercept	logical. Should an intercept be included in the <i>null</i> model?
object	an object inheriting from class "glm".
type	character, partial matching allowed. Type of weights to extract from the fitted model object. Can be abbreviated.
...	For <code>glm</code> : arguments to be used to form the default <code>control</code> argument if it is not supplied directly.

For `weights`: further arguments passed to or from other methods.

An object of class "glm" is a list containing at least the following components:

coefficients	a named vector of coefficients
residuals	the <i>working</i> residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are NA.
fitted.values	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
rank	the numeric rank of the fitted linear model.
family	the <code>family</code> object used.
linear.predictors	the linear fit on link scale.
deviance	up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
aic	A version of Akaike's <i>An Information Criterion</i> , minus twice the maximized log-likelihood plus twice the number of parameters, computed via the <code>aic</code> component of the family. For binomial and Poisson families the dispersion is fixed at one and the number of parameters is the number of coefficients. For gaussian, Gamma and inverse gaussian families the dispersion is estimated from the residual deviance, and the number of parameters is the number of coefficients

plus one. For a gaussian family the MLE of the dispersion is used so this is a valid value of AIC, but for Gamma and inverse gaussian families it is not. For families fitted by quasi-likelihood the value is NA.	
null.deviance	The deviance for the null model, comparable with <code>deviance</code> . The null model will include the offset, and an intercept if there is one in the model. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.
iter	the number of iterations of IWLS used.
weights	the <i>working</i> weights, that is the weights in the final iteration of the IWLS fit.
prior.weights	the weights initially supplied, a vector of 1s if none were.
df.residual	the residual degrees of freedom.
df.null	the residual degrees of freedom for the null model.
y	if requested (the default) the <code>y</code> vector used. (It is a vector even for a binomial model.)
x	if requested, the model matrix.
model	if requested (the default), the model frame.
converged	logical. Was the IWLS algorithm judged to have converged?
boundary	logical. Is the fitted value on the boundary of the attainable values?
call	the matched call.
formula	the formula supplied.
terms	the <code>terms</code> object used.
data	the <code>data</code> argument.
offset	the offset vector used.
control	the value of the <code>control</code> argument used.
method	the name of the fitter function used (when provided as a <code>character</code> string to <code>glm()</code> ) or the fitter <code>function</code> (when provided as that).
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting.
na.action	(where relevant) information returned by <code>model.frame</code> on the special handling of NAs.

## Тема 7. Метод опорних векторів

Метод опорних векторів (support vector), званий раніше алгоритмом «узагальненого портрета», був розроблений В. Вапніком та А. Червоненкісом 1974 року.

Основна ідея класифікатора на опорних векторах полягає в тому, щоб будувати розподілячу поверхню з використанням невеликої підмножини точок, що перебуватимуть у зоні, критичної для поділу, тоді як інші спостереження навчальної вибірки ігноруються (є «резервуаром» для оптимізаційного алгоритму).

Метод опорних векторів будує функцію класифікації  $F$  у вигляді

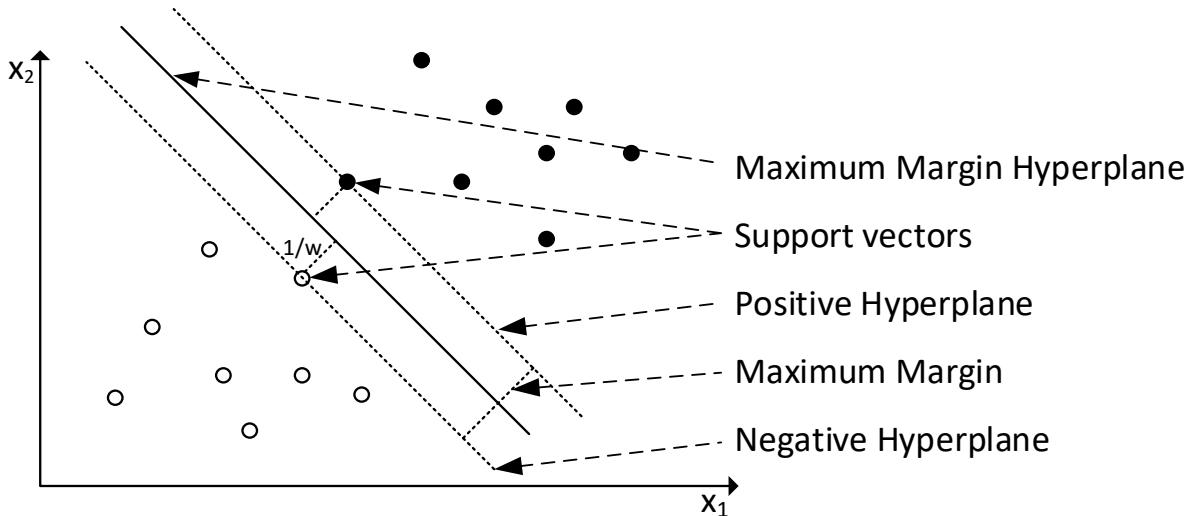
$$F(x) = \text{sign} (\langle w, x \rangle + b)$$

де  $w$  – нормальній вектор, що розподіляє гіперплощини,  $b$  – допоміжний параметр. Ті об'єкти, для яких  $F(x) = 1$  потрапляють до одного класу, об'єкти, для яких  $F(x) = -1$  – до іншого.

Необхідно знайти такі  $w$  і  $b$ , що максимізують відстань до кожного класу –  $\frac{1}{\|w\|}$ . Запишемо це у вигляді завдання оптимізації, що є стандартною задачею квадратичного програмування і вирішується за допомогою множників Лагранжа (пошук мінімума  $\frac{1}{\|w\|}$  еквівалентний пошуку максимума  $\|w\|$ )

$$\begin{cases} \arg \max \|w\|^2, \\ y_i(\langle w, x_i \rangle + b) \geq 1 \end{cases}$$

Опорними векторами називаються спостереження, що перебувають безпосередньо на кордоні розподіляючої смуги або на неправильному для свого класу боці щодо кордонів зазору.



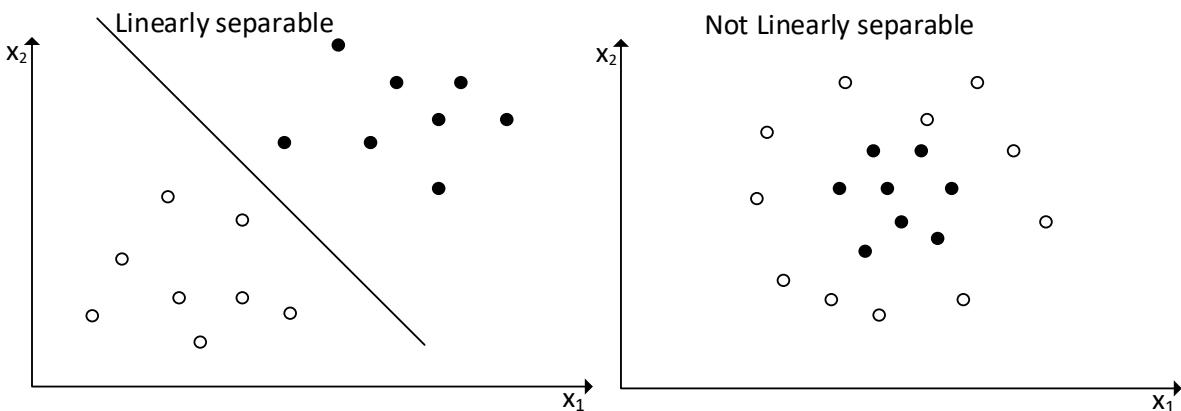
Якщо є два класи спостережень і передбачається лінійна форма кордону між класами, то можливі два випадки:

1. У разі лінійної розподільності спостережень можлива побудова гіперплощини  $F(x) = \sum w_i x_i + b$ . Оскільки таких гіперплощин може бути безліч, то оптимальною є така поверхня, яка максимально віддалена від навчальних точок, тобто має максимальний проміжок  $M$  (margin);
2. В іншому випадку, хмари точок перекриваються і обидва класи лінійно нерозподільні. Оптимальну розподіляючу гіперплощину такого класифікатора  $F(x) = \sum w_i x_i + b$  також знаходить з умови максимізації ширини проміжку  $M$ , але при цьому дозволяється невірно класифікувати деяку невелику групу спостережень, що належать до опорних векторів. Для цього задається додаткова умова оптимізації, допустима кількість порушень кордону проміжку та їх вираженість, що зазвичай вибирається з використанням перехресної перевірки.

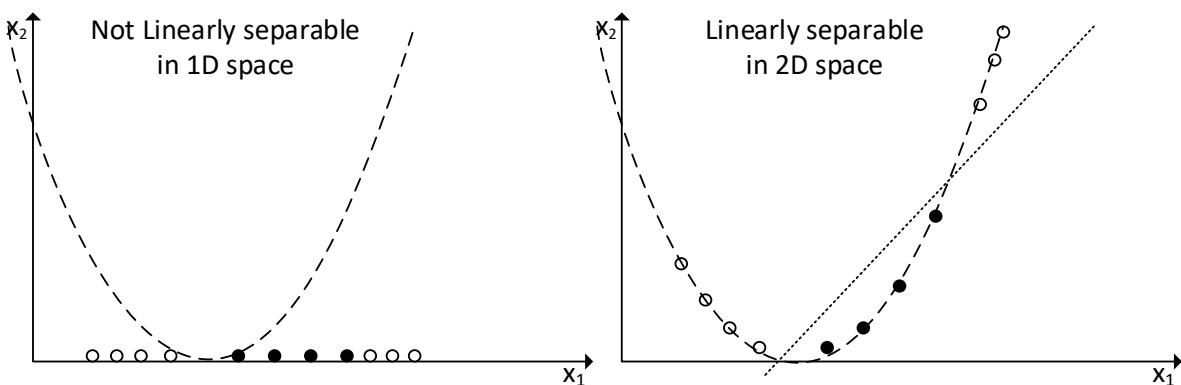
Математично пошук рішення зводиться до задачі квадратичної оптимізації з лінійними обмеженнями, що гарантовано сходиться до одного глобального мінімуму.

Оскільки на розташування гіперплощини впливають тільки ті спостереження, що перебувають на кордонах проміжку або порушують його, то вирішальне правило такого класифікатора є досить стійким до викидів більшості точок, розташованих поза «критичної зони» поділу. Ця властивість відрізняє його від інших класифікаторів.

За наявності нелінійного зв'язку між ознаками і відгуком якість лінійних класифікаторів може виявитися незадовільною.



Для подолання проблеми нелінійності елементи навчальної вибірки вкладаються в простір  $\mathbf{X}$  більш високої розмірності з допомогою відображення  $\varphi(\mathbf{X})$ .

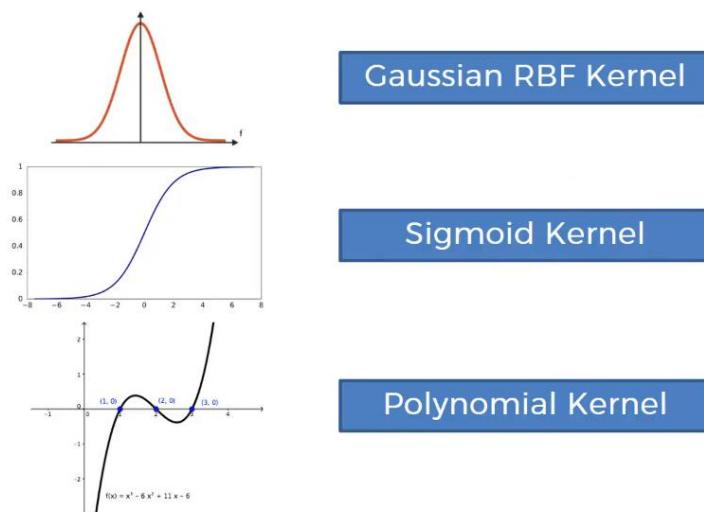


Водночас відображення  $\varphi$  вибирається так, щоб в новому просторі  $\mathbf{X}$  вибірка була лінійно розподільна. Класифікатор набуває вигляду

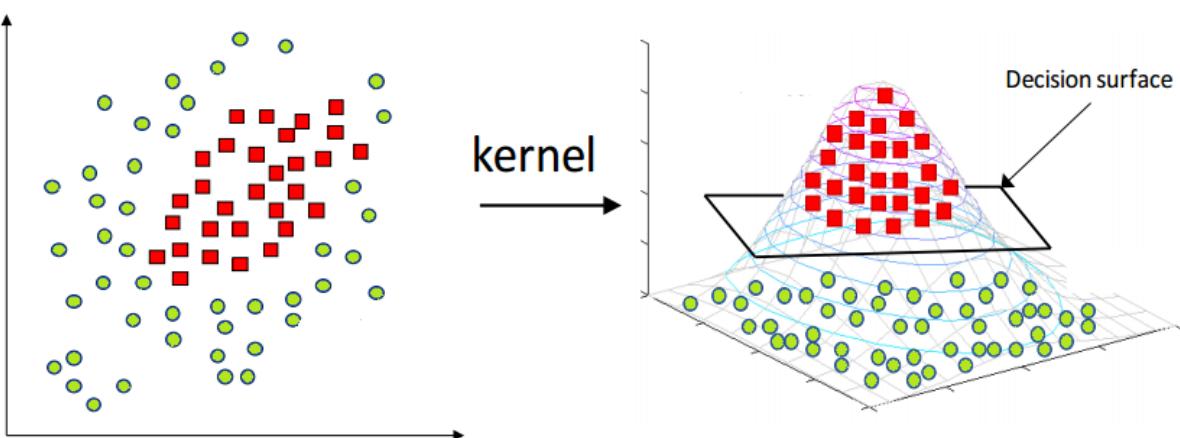
$$F(x) = \text{sign} (\langle w, \varphi(x) \rangle + b)$$

Вираз  $k(x, x') = \langle \varphi(x), \varphi(x') \rangle$  називається ядром класифікатора. З математичної точки зору ядром може бути будь-яка позитивно визначена симетрична функція двох змінних. Позитивна визначеність необхідна для того, щоб відповідна функція Лагранжа в задачі оптимізації була обмежена знизу, тобто задача оптимізації була б коректно визначена. Точність класифікатора залежить, зокрема, від вибору ядра.

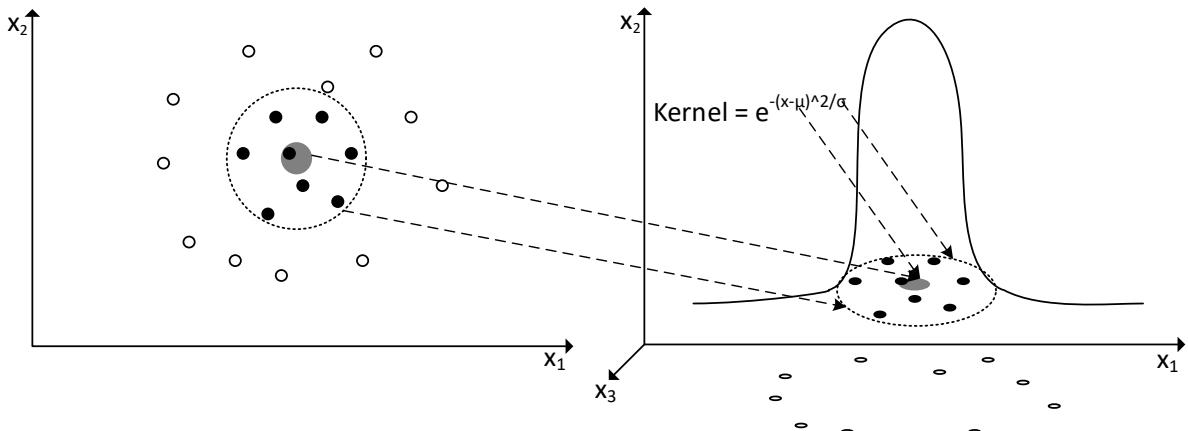
Найчастіше на практиці зустрічаються такі ядра



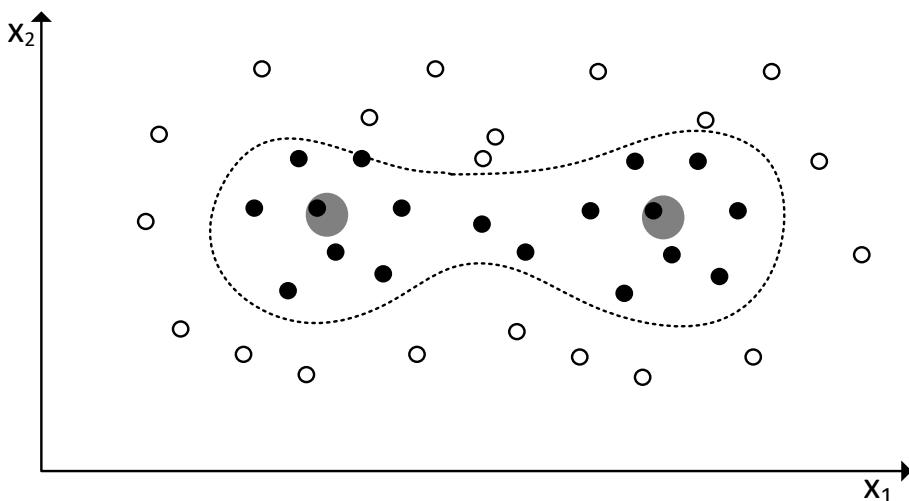
Обчислення екстремуму в розширених просторах величезних розмірностей виявилося можливим тому, що ядро формується тільки для обмеженого набору опорних векторів.



Як було сказано вище, кожне ядро характеризується параметрами, що підлягають оптимізації, наприклад  $\mu$  та  $\sigma$ .



Сукупність ядер дозволяє будувати моделі з використанням розподіляючих поверхонь різної форми.



Отже, машину опорних векторів SVM (Support Vector Machine) можна розглядати як нелінійне узагальнення лінійного класифікатора, засноване на розширенні розмірності вихідного простору предикторів за допомогою спеціальних ядерних функцій.

## Лабораторна робота 6

### Support Vector Machine (SVM)

#### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
```

Висновок: для побудови моделі банківського скрингу використані дані про наявність прострочених платежів за кредитом.

#### Features Scaling

```
f <- f[,c('age','income','delays')]
sc <- scale(f[-3])
f$age <- sc[,c('age')]
f$income <- sc[,c('income')]
```

Висновок: залишимо для моделювання кількісні змінні, адже лише вони виявилися значущими. Проведемо шкалювання кількісних змінних.

#### Splitting the scaled dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f$delays, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

Висновок: підготований датасет розподілено на навчальну та тестову вибірки.

#### Fitting linear model

```
# install.packages('e1071')
library(e1071)
class_svm_1 = svm(delays ~ ., data = f_train, kernel = 'linear')
summary(class_svm_1)

##
## Call:
## svm(formula = delays ~ ., data = f_train, kernel = "linear")
##
##
## Parameters:
##   SVM-Type: C-classification
##   SVM-Kernel: linear
##   cost: 1
```

```
##  
## Number of Support Vectors: 133  
##  
## ( 66 67 )  
##  
##  
## Number of Classes: 2  
##  
## Levels:  
## NO YES
```

Висновок: для навчання базової моделі, заснованої на методі опорних векторів, вибрано лінійне ядро.

## Predicting

```
y <- predict(class_svm_1, f_test[, c('age', 'income')])
```

Висновок: визначено класи об'єктів (вектор y).

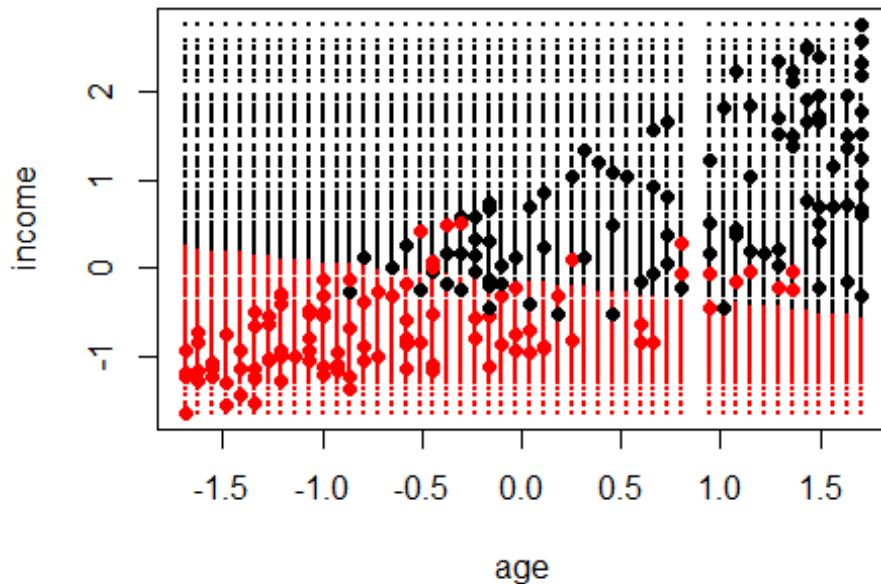
## Confusion Matrix

```
cm = table(f_test[, 'delays'], y)  
print(cm)  
  
##      y  
##      NO YES  
##  NO  86  12  
##  YES 14  88
```

Висновок: точність моделі –  $(86+88) / 200 = 87\%$ , частка невірно класифікованих випадків –  $(12+14) / 200 = 13\%$ . Чутливість –  $88 / (14+88) = 86\%$ , специфічність –  $86 / (86+12) = 88\%$ , тобто модель більш чутлива до виявлення негативних випадків. У цьому разі – кредиторів, що не мають залогованисті.

## Visualising the Test set results

```
xgrid = expand.grid(age = f_test$age, income = f_test$income)  
ygrid = predict(class_svm_1, xgrid)  
  
#Finally, you plot the points and color them according to the decision boundary. You can see that the decision boundary is linear. You can put the data points in the plot as well to see where they lie.  
  
plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)  
points(f_test[, c('age', 'income')], col = as.factor(f_test$delays), pch = 19)
```



**Висновок:** на графіку світлим позначені випадки затримки з повернення кредиту, темним – хороши кредитори. Червоним виділена зона високої ймовірності неповернення кредиту. Модель описує лінійний варіант розподіляючої кривої.

### Fitting RBF model

```
# install.packages('e1071')
library(e1071)
class_svm_r = svm(delays ~ ., data = f_train, kernel = 'radial')
summary(class_svm_r)

##
## Call:
## svm(formula = delays ~ ., data = f_train, kernel = "radial")
##
##
## Parameters:
##   SVM-Type: C-classification
##   SVM-Kernel: radial
##   cost: 1
##
## Number of Support Vectors: 137
##
## ( 68 69 )
##
##
## Number of Classes: 2
##
## Levels:
## NO YES
```

**Висновок:** для навчання моделі, заснованої на методі опорних векторів, вибрано нелінійне ядро.

## Predicting

```
y <- predict(class_svm_r, f_test[, c('age', 'income')])
```

Висновок: визначені класи об'єктів (вектор y).

## Confusion Matrix

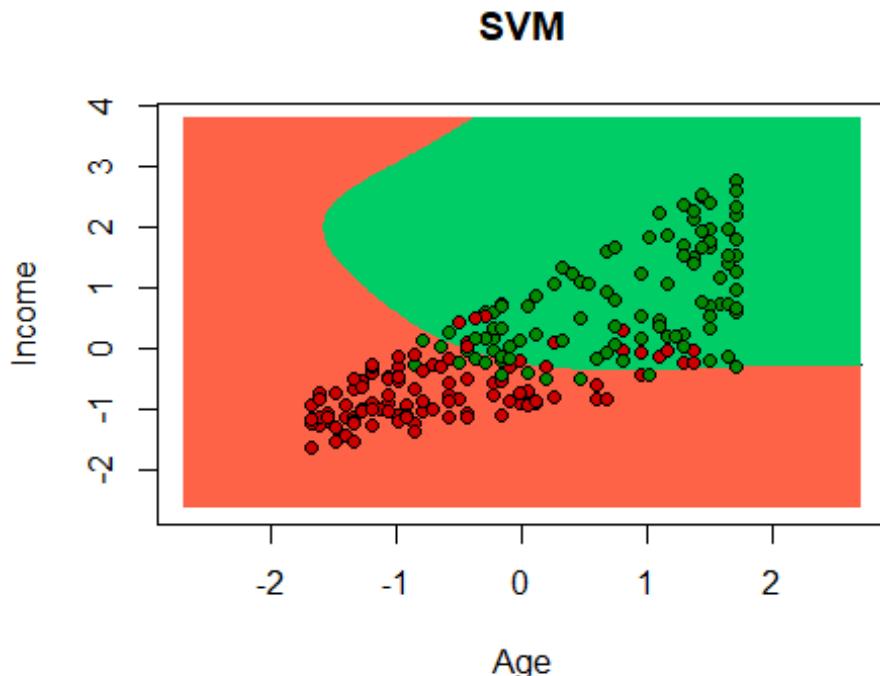
```
cm = table(f_test[, 'delays'], y)
print(cm)

##          y
##      NO YES
##    NO  85 13
##    YES 15 87
```

Висновок: точність моделі погіршилася –  $(85+87) / 200 = 86\%$ , частка невірно класифікованих випадків –  $(13+15) / 200 = 14\%$ . Чутливість –  $87 / (15+87) = 85\%$ , специфічність –  $85 / (85+13) = 87\%$ , тобто модель більш чутлива до виявлення негативних випадків. У цьому разі – кредиторів, що не мають заборгованості.

## Visualising the Test set results

```
#install.packages("ElemStatLearn")
library(ElemStatLearn)
set = f_test[,c('age', 'income', 'delays')]
X1 = seq(min(set['age']) - 1, max(set['age']) + 1, by = 0.01)
X2 = seq(min(set['income']) - 1, max(set['income']) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('age', 'income')
y_grid = predict(class_svm_r, grid_set)
plot(set[, -3],
     main = 'SVM',
     xlab = 'Age', ylab = 'Income',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 'YES', 'tomato',
'springgreen3'))
points(set, pch = 21, bg = ifelse(set[, 3] == 'YES', 'red3', 'green4'))
```



**Висновок:** на графіку червоним позначені випадки затримки з повернення кредиту, зеленим – хороши кредитори. Червоним виділена зона високої ймовірності неповернення кредиту. Модель описує нелінійний варіант розподіляючої кривої.

### Довідка: svm

```
## S3 method for class 'formula'
svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)
## Default S3 method:
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,
..., subset, na.action = na.omit)
```

### Arguments

formula	a symbolic description of the model to be fit.
Data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which ‘svm’ is called from.
x	a data matrix, a vector, or a sparse matrix (object of class <a href="#">Matrix</a> provided by the Matrix package, or of class <a href="#">matrix.csr</a> provided by the SparseM package, or of class <a href="#">simple_triplet_matrix</a> provided by the slam package).
y	a response vector with one label for each row/component of x. Can be either a factor (for classification tasks) or a numeric vector (for regression).

Scale	A logical vector indicating the variables to be scaled. If <code>scale</code> is of length 1, the value is recycled as many times as needed. Per default, data are scaled internally (both <code>x</code> and <code>y</code> variables) to zero mean and unit variance. The center and scale values are returned and used for later predictions.
Type	<code>svm</code> can be used as a classification machine, as a regression machine, or for novelty detection. Depending of whether <code>y</code> is a factor or not, the default setting for <code>type</code> is C-classification or eps-regression, respectively, but may be overwritten by setting an explicit value. Valid options are:
	<ul style="list-style-type: none"> <li>• C-classification</li> <li>• nu-classification</li> <li>• one-classification (for novelty detection)</li> <li>• eps-regression</li> <li>• nu-regression</li> </ul>
Kernel	the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type.
linear:	
	$u' * v$
polynomial:	
	$(gamma * u' * v + coef0)^{degree}$
radial basis:	
	$exp(-gamma *  u - v ^2)$
sigmoid:	
	$tanh(gamma * u' * v + coef0)$
Degree	parameter needed for kernel of type polynomial (default: 3)
Gamma	parameter needed for all kernels except linear (default: 1/(data dimension))
coef0	parameter needed for kernels of type polynomial and sigmoid (default: 0)
Cost	cost of constraints violation (default: 1). It is the ‘C’-constant of the regularization term in the Lagrange formulation.
Nu	parameter needed for nu-classification, nu-regression, and one-classification
class.weights	a named vector of weights for the different classes, used for asymmetric class sizes. Not all factor levels have to be supplied (default weight: 1). All components have to be named. Specifying "inverse" will choose the weights inversely proportional to the class distribution.
cachesize	cache memory in MB (default 40)
tolerance	tolerance of termination criterion (default: 0.001)

<code>epsilon</code>	epsilon in the insensitive-loss function (default: 0.1)
<code>shrinking</code>	option whether to use the shrinking-heuristics (default: TRUE)
<code>Cross</code>	if a integer value $k > 0$ is specified, a $k$ -fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification and the Mean Squared Error for regression
<code>Fitted</code>	logical indicating whether the fitted values should be computed and included in the model or not (default: TRUE)
<code>probability</code>	logical indicating whether the model should allow for probability predictions.
<code>...</code>	additional parameters for the low level fitting function <code>svm.default</code>
<code>Subset</code>	An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
<code>na.action</code>	A function to specify the action to be taken if <code>NAs</code> are found. The default action is <code>na.omit</code> , which leads to rejection of cases with missing values on any required variable. An alternative is <code>na.fail</code> , which causes an error if <code>NA</code> cases are found. (NOTE: If given, this argument must be named.)

An object of class "`svm`" containing the fitted model, including:

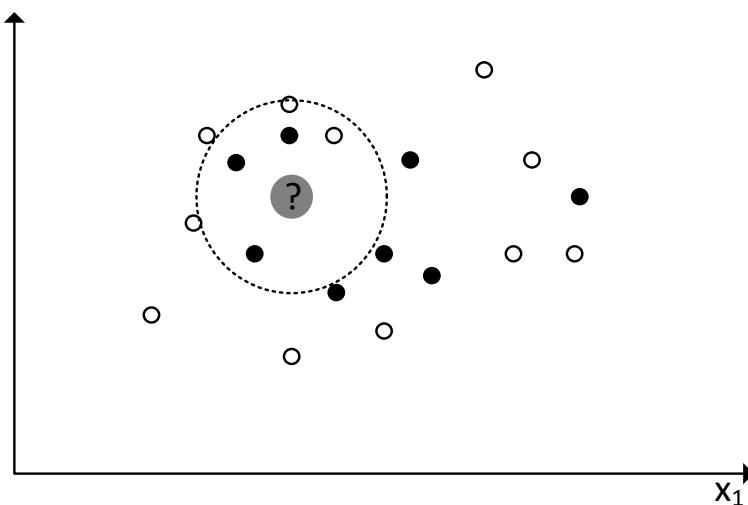
<code>SV</code>	The resulting support vectors (possibly scaled).
<code>Index</code>	The index of the resulting support vectors in the data matrix. Note that this index refers to the preprocessed data (after the possible effect of <code>na.omit</code> and <code>subset</code> )
<code>Coefs</code>	The corresponding coefficients times the training labels.
<code>Rho</code>	The negative intercept.
<code>Sigma</code>	In case of a probabilistic regression model, the scale parameter of the hypothesized (zero-mean) laplace distribution estimated by maximum likelihood.
<code>probA,</code> <code>probB</code>	numeric vectors of length $k$ ( $k - 1$ ) / 2, $k$ number of classes, containing the parameters of the logistic distributions fitted to the decision values of the binary classifiers ( $1 / (1 + \exp(a x + b))$ ).

### Тема 8. Метод k-найближчих сусідів

Основою kNN-класифікатора є гіпотеза компактності, що передбачає: тестований об'єкт  $d$  матиме таку ж мітку класу, як і навчальні об'єкти в локальній області його найближчого оточення. У варіанті 1NN аналізований об'єкт належить до певного класу в залежності від інформації про його найближчого сусіда. У варіанті kNN кожен об'єкт належить до переважного класу найближчих сусідів, де  $k$  – параметр алгоритму.

Вирішальні правила в методі kNN визначаються межами суміжних сегментів діаграми Вороного (Voronoi tessellation), що розподіляє площину на  $n$  опуклих багатокутників, кожен з яких містить один і тільки один об'єкт навчальної вибірки. В  $n$ -мірних просторах кордони рішень складаються з сегментів  $(n-1)$ -мірних напівплощин, утворених опуклими багатогранниками Вороного.

Алгоритм будується за принципом «більшості голосів», тобто як результат оголошується мітка класу-переможця. На рис. тестований об'єкт при  $k = 5$  буде віднесений до класу «чорних».

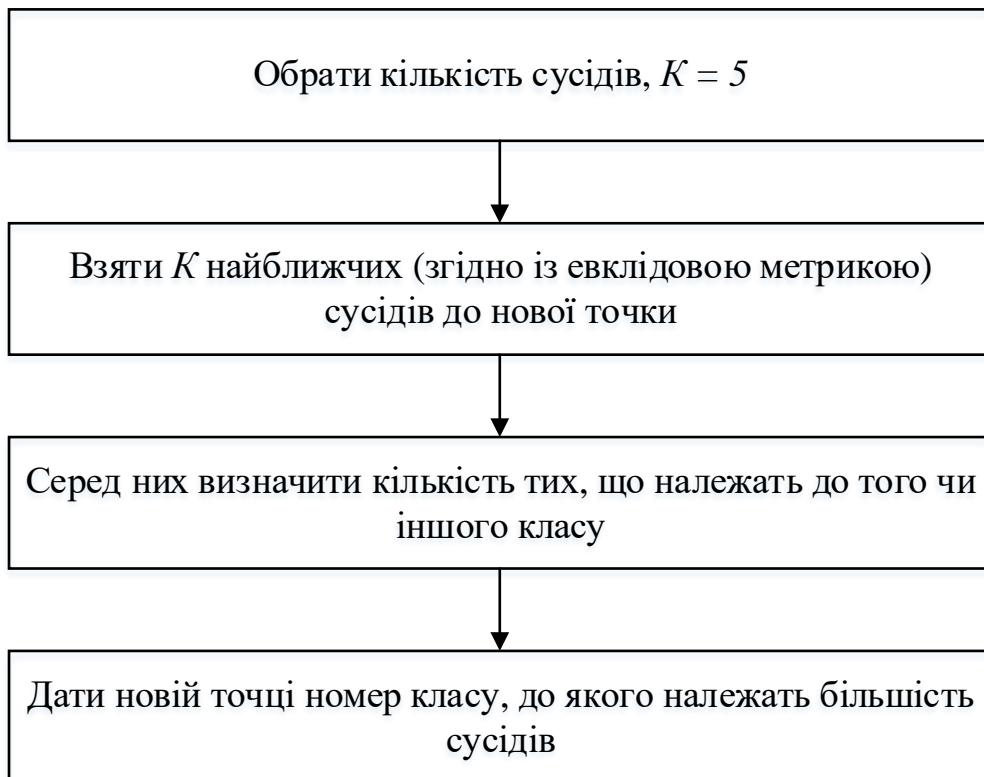


Імовірнісний варіант методу kNN використовує для ранжирування передбачуваних класів суму «голосів» сусідів з урахуванням їх ваг, зокрема, евклідової відстані між тестованим об'єктом і кожним із сусідів.

Варіант 1NN завжди забезпечує 100 % правильне розпізнавання прикладів навчальної вибірки (найближчий сусід – він сам), проте часто помилляється на невідомих йому даних. Зі збільшенням  $k > 1$  до деяких меж якість розпізнавання на контрольній вибірці буде зростати.

Оптимальне в сенсі точності прогнозів значення  $k$  може бути знайдено з використанням перехресної перевірки. Для цього за фіксованим значенням  $k$  будується модель  $k$ -найближчих сусідів і оцінюється помилка класифікації. Ці дії повторюються для різних  $k$ ; значення, що відповідає найменшій помилці, є оптимальним.

Алгоритм методу К найближчих сусідів:



## Лабораторна робота 7

### K-Nearest Neighbors (K-NN)

#### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
```

Висновок: для побудови моделі банківського скорингу використані дані про наявність прострочених платежів за кредитом.

#### Features Scaling

```
f <- f[,c('age','income','delays')]
sc <- scale(f[-3])
f$age <- sc[,c('age')]
f$income <- sc[,c('income')]
head (f)

##           age      income delays
## 1  0.3885629 -0.7735227    YES
## 2 -0.1660318  0.1985406     NO
## 3  0.5965360 -0.8487661    YES
## 4 -1.3445456 -0.5541803    YES
## 5  1.0124820  1.7870712     NO
## 6  1.0124820  0.8020151     NO
```

Висновок: залишимо для моделювання кількісні змінні, адже лише вони виявилися значущими. Проведемо шкалювання кількісних змінних.

#### Splitting the scaled dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f$delays, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

Висновок: підготований датасет розподілено на навчальну та тестову вибірки.

#### Fitting & predicting

```
library(class)
y = knn(train = f_train[,c('age','income')],
        test = f_test[,c('age','income')],
        cl = f_train[, 'delays'],
        k = 5,
        prob = TRUE)
```

**Висновок:** і навчання, і прогнозування за моделлю к найближчих сусідів здійснюється однією функцією. У результаті отримуємо вектор класів об'єктів.

## Confusion Matrix

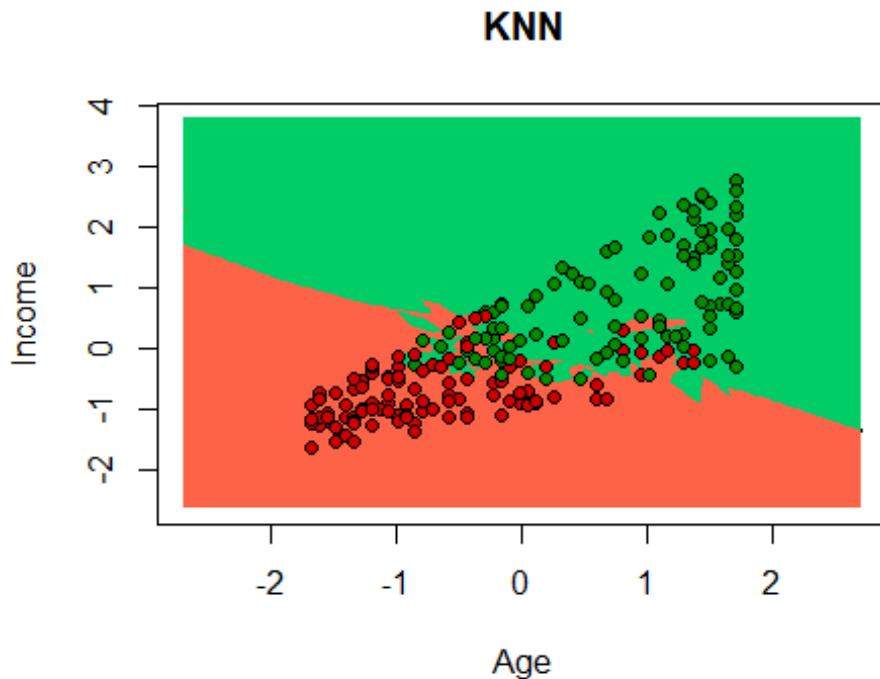
```
cm = table(f_test[, 'delays'], y == 'YES')
print(cm)

##
##      FALSE  TRUE
##  NO    81   17
##  YES   11   91
```

**Висновок:** точність моделі –  $(81+91) / 200 = 86\%$ , частка невірно класифікованих випадків –  $(17+11) / 200 = 14\%$ . Чутливість –  $91 / (11+91) = 89\%$ , специфічність –  $81 / (81+17) = 83\%$ , тобто модель більш чутлива до виявлення позитивних випадків. У цьому разі – кредиторів, що мають заборгованість.

## Visualising the Test set results

```
#install.packages("ElemStatLearn")
library(ElemStatLearn)
set = f_test
X1 = seq(min(set['age']) - 1, max(set['age']) + 1, by = 0.01)
X2 = seq(min(set['income']) - 1, max(set['income']) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('age', 'income')
y_grid = knn(train = f_train[,c('age','income')], test = grid_set, cl =
f_train[, 'delays'], k = 5)
plot(set[, -3],
      main = 'KNN',
      xlab = 'Age', ylab = 'Income',
      xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 'YES', 'tomato',
'springgreen3'))
points(set, pch = 21, bg = ifelse(set[, 3] == 'YES', 'red3', 'green4'))
```



**Висновок:** на графіку червоним позначені випадки затримки з повернення кредиту, зеленим – хороши кредитори. Червоним виділена зона високої ймовірності неповернення кредиту. Модель описує нелінійний варіант розподіляючої кривої.

### Довідка: knn

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

#### Arguments

- train matrix or data frame of training set cases.
- test matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
- cl factor of true classifications of training set
- k number of neighbours considered.
- l minimum vote for definite decision, otherwise doubt. (More precisely, less than  $k-l$  dissenting votes are allowed, even if  $k$  is increased by ties.)
- prob If this is true, the proportion of the votes for the winning class are returned as attribute prob.
- use.all controls handling of ties. If true, all distances equal to the  $k$ th largest are included. If false, a random selection of distances equal to the  $k$ th is chosen to use exactly  $k$  neighbours.

## Тема 9. Байєсова класифікація

Наївний байєсовський класифікатор (Naive Bayes Classifier) – ймовірнісний класифікатор, заснований на теоремі Байєса з нежорсткими припущеннями про незалежність подій<sup>3</sup>, вона задає формальний метод, що дозволяє в процесі ухвалення рішень врахувати нову інформацію.

Основою байєсівської класифікації є гіпотеза максимальної ймовірності, тобто вважається, що об'єкт  $A$  належить класу  $H_i$ , якщо досягається найбільша апостеріорна ймовірність:  $\max \{P(H_i | A)\}$ .

За формулою Байєса

$$P(H_i | A) = \frac{P(A|H_i)P(H_i)}{P(A)}$$

де  $P(A|H_i)$  – ймовірність зустріти об'єкт  $A$  серед об'єктів класу  $H_i$ ;  $P(H_i)$  та  $P(A)$  – априорні ймовірності класу  $H_i$  та об'єкта  $A$  (остання, не впливає на вибір класу, нею можна знехтувати).

### <sup>3</sup> Примітки

Подія  $A$  називається незалежною від події  $B$ , якщо ймовірність події  $A$  не залежить від того, чи відбулася подія  $B$  чи ні.

Сумою кількох подій називається подія, що містить появу хоча б однієї із цих подій.

Імовірність суми двох незалежних подій дорівнює сумі ймовірностей цих подій

$$P(A + B) = P(A) + P(B).$$

Якщо якісь події утворюють повну групу подій, то сума їх ймовірностей дорівнює 1.

Добудком кількох подій називається подія, що містить спільну появу всіх цих подій.

Імовірність добутку двох незалежних подій дорівнює добутку їх ймовірностей.

Імовірність добутку двох залежних подій дорівнює добутку ймовірності однієї з них та умовної ймовірності іншої за наявністю першої

$$P(AB) = P(A)P(B | A) = P(B)P(A | B).$$

Формула повної ймовірності

$$P(A) = \sum_{i=1}^n P(A|H_i)P(H_i)$$

Якщо зробити «найвне» припущення, що всі ознаки, які описують об'єкти, що класифікуються, абсолютно рівноправні між собою та не пов'язані один з одним, то  $P(A|H_i)$  можна обчислити як добуток ймовірностей зустріти ознаку  $A_j$  серед об'єктів класу  $H_i$

$$P(A|H_i) = \prod_j P(A_j|H_i)$$

де  $P(A_j|H_i)$  – імовірнісна оцінка вкладу ознаки  $A_j$  в те, що  $A \in H_i$ .

Розглянемо приклад бінарної класифікації. Нехай є два класи людей, одні склонні до кредитування (YES), другі – ні (NO). І нехай вони описуються двома ознаками – віком і доходом. Необхідно долучити нову людину, дані про вік і доходи якої відомі, до одного з класів. Скористаємося формулою Байєса

$$P(NO|X) = \frac{P(X|NO)P(NO)}{P(X)}$$

де  $P(NO)$  – априорна ймовірність;

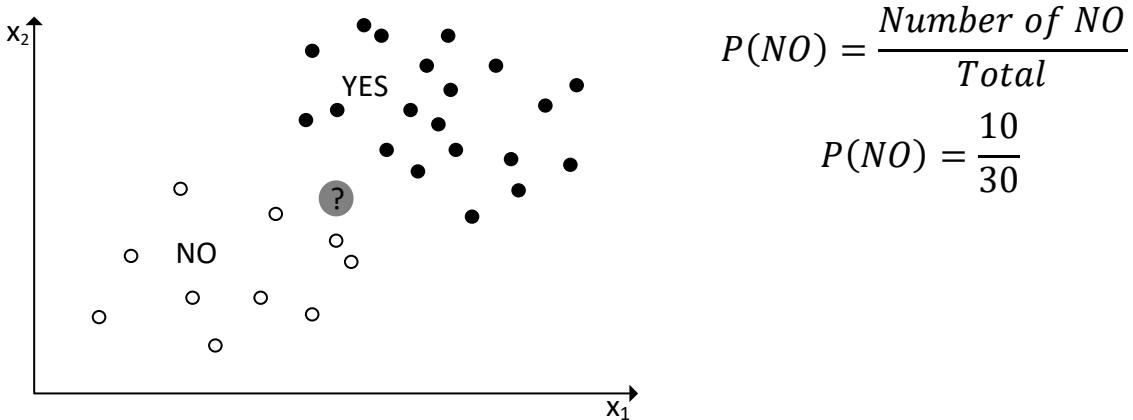
$P(X|NO)$  – умовна ймовірність;

$P(X)$  – гранична ймовірність ;

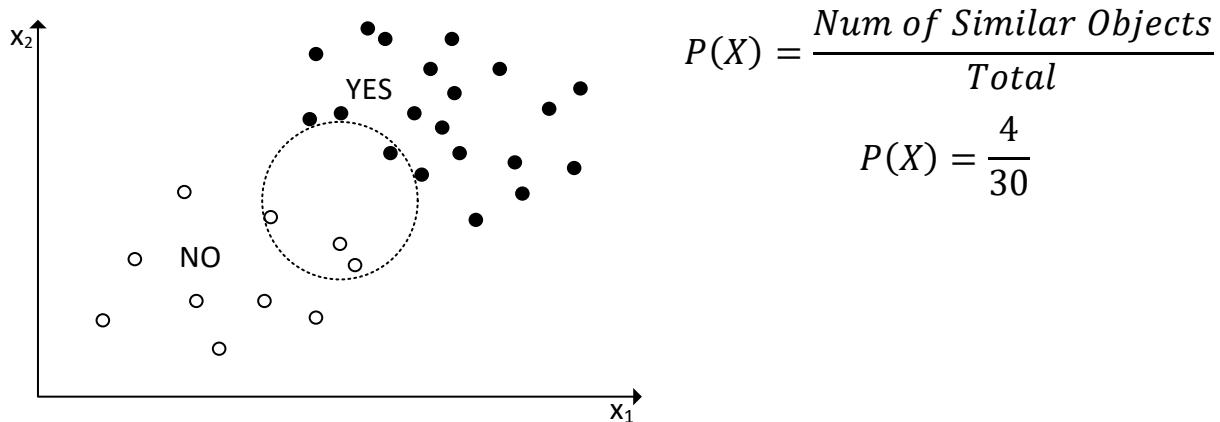
$P(NO|X)$  – апостеріорна ймовірність.

Порядок розрахунків за моделлю найвного байєсівського класифікатора такий:

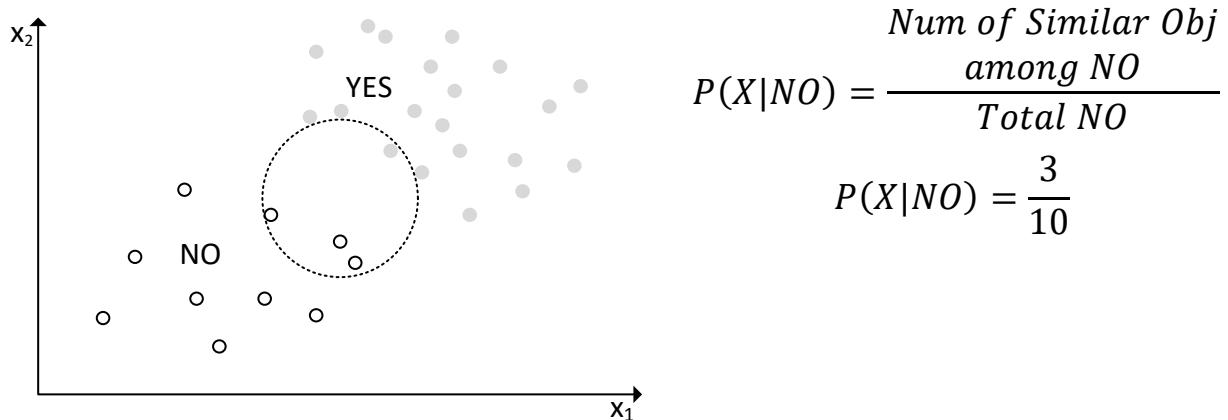
1. По-перше обчислюється априорна ймовірність для кожного з класів



2. Потім обчислюється умовна ймовірність



3. Потім обчислюється гранична ймовірність

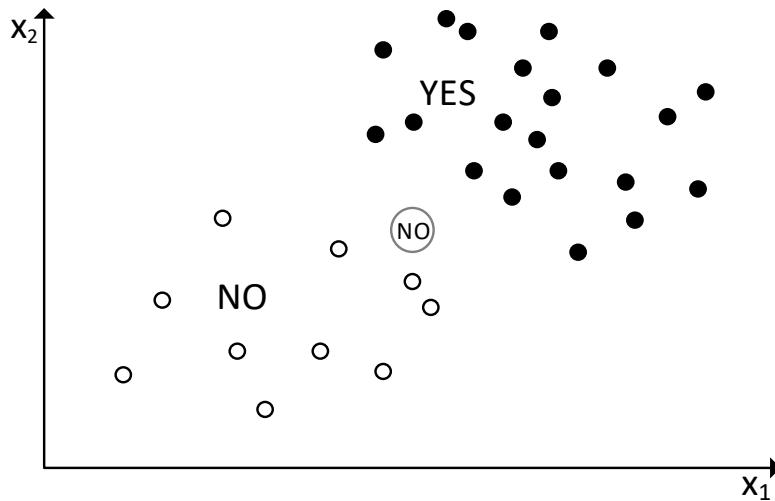


4. Потім обчислюється апостеріорна ймовірність для кожного з класів

$$P(NO|X) = \frac{\frac{3}{10} \cdot \frac{10}{30}}{\frac{10}{4} \cdot \frac{30}{30}} = 0,75$$

$$P(YES|X) = \frac{\frac{1}{20} \cdot \frac{20}{30}}{\frac{20}{4} \cdot \frac{30}{30}} = 0,25$$

5. Новий об'єкт належить до класу, що відповідає максимальній апостеріорній ймовірності –  $P(NO|X) \geq P(YES|X)$



На практиці під час множення дуже малих умовних ймовірностей може спостерігатися втрата значущих розрядів, в зв'язку з чим замість самих оцінок ймовірностей  $P(A_j|H_i)$  застосовують логарифми цих ймовірностей. Оскільки логарифм – монотонно зростаюча функція, то клас  $H_i$  з найбільшим значенням логарифма ймовірності залишиться найбільш імовірним. При цьому важливо, щоб значення ймовірностей були не надто близькі до 0. Тоді вирішальне правило найвного байєсівського класифікатора набуває такого вигляду

$$H = \max_i \left\{ \log P(H_i) + \sum_j P(A_j|H_i) \right\}$$

Залежно від природи ймовірнісної моделі, байєсівські класифікатори можуть навчатися дуже ефективно. Перевагою найвного байєсівського класифікатора є мала кількість даних, необхідних для навчання, оцінки параметрів і класифікації.

## Лабораторна робота 8

### Naive Bayes

#### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
```

Висновок: для побудови моделі банківського скорингу використані дані про наявність прострочених платежів по кредиту.

#### Features Scaling

```
f <- f[,c('age','income','delays')]
sc <- scale(f[ -3])
f$age <- sc[,c('age')]
f$income <- sc[,c('income')]
```

Висновок: залишимо для моделювання лише кількісні змінні, адже лише вони виявилися значущими. Проведемо шкалювання кількісних змінних.

#### Splitting the scaled dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f$delays, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

Висновок: підготований датасет розподілено на навчальну та тестову вибірки.

#### Fitting

```
# install.packages('e1071')
library(e1071)
class_nb = naiveBayes(delays ~ ., data = f_train)
```

Висновок: для навчання моделі використано функцію `naiveBayes`.

#### Predicting

```
y <- predict(class_nb, f_test[, c('age','income')])
```

Висновок: визначено класи об'єктів (вектор  $y$ ).

## Confusion Matrix

```
cm = table(f_test[, 'delays'], y)
print(cm)

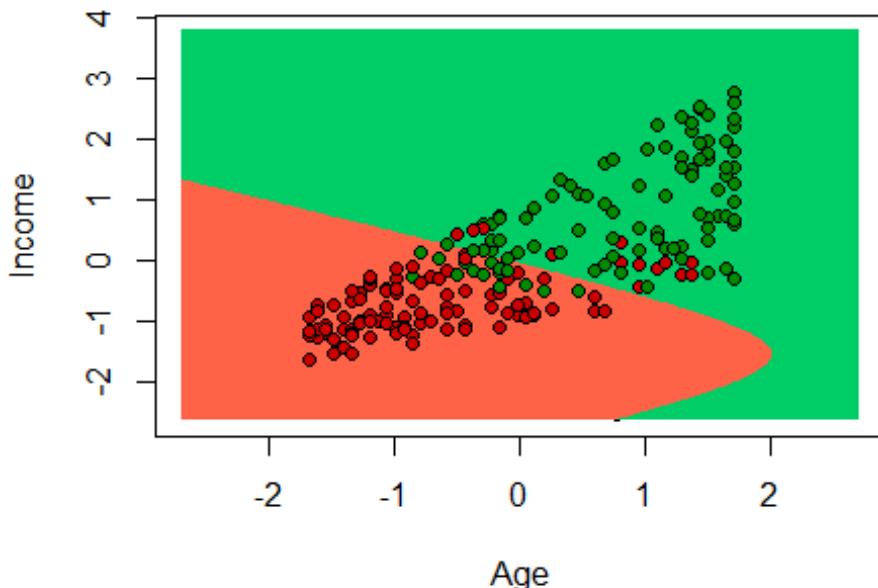
##      y
##    NO YES
##  NO  83 15
##  YES 13 89
```

Висновок: точність моделі –  $(83+89) / 200 = 86\%$ , частка неправильно класифікованих випадків –  $(13+15) / 200 = 14\%$ . Чутливість –  $89 / (13+89) = 87\%$ , специфічність –  $83 / (83+15) = 85\%$ , тобто модель більш чутлива до виявлення позитивних випадків. У цьому разі – кредиторів, що мають заборгованість.

## Visualising the Test set results

```
#install.packages("ElemStatLearn")
library(ElemStatLearn)
set = f_test[,c('age','income','delays')]
X1 = seq(min(set['age']) - 1, max(set['age']) + 1, by = 0.01)
X2 = seq(min(set['income']) - 1, max(set['income']) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('age', 'income')
y_grid = predict(class_nb, grid_set)
plot(set[, -3],
      main = 'Naive Bayes',
      xlab = 'Age', ylab = 'Income',
      xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 'YES', 'tomato',
'springgreen3'))
points(set, pch = 21, bg = ifelse(set[, 3] == 'YES', 'red3', 'green4'))
```

## Naive Bayes



**Висновок:** на графіку червоним позначені випадки затримки з поверненням кредиту, зеленим – хороши кредитори. Червоним виділена зона високої ймовірності неповернення кредиту. Модель описує нелінійний варіант розподіляючої кривої.

### Довідка: naiveBayes

```
## S3 method for class 'formula'
naiveBayes(formula, data, laplace = 0, ..., subset, na.action = na.pass)
## Default S3 method:
naiveBayes(x, y, laplace = 0, ...)

## S3 method for class 'naiveBayes'
predict(object, newdata,
        type = c("class", "raw"), threshold = 0.001, eps = 0, ...)
```

#### Arguments

- `x` A numeric matrix, or a data frame of categorical and/or numeric variables.
- `y` Class vector.
- `formula` A formula of the form  $\text{class} \sim x_1 + x_2 + \dots$ . Interactions are not allowed.
- `data` Either a data frame of predictors (categorical and/or numeric) or a contingency table.
- `laplace` positive double controlling Laplace smoothing. The default (0) disables Laplace smoothing.
- `...` Currently not used.

subset	For data given in a data frame, an index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
na.action	A function to specify the action to be taken if NAs are found. The default action is not to count them for the computation of the probability factors. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)
object	An object of class "naiveBayes".
newdata	A data frame with new predictors (with possibly fewer columns than the training data). Note that the column names of newdata are matched against the training data ones.
type	If "raw", the conditional a-posterior probabilities for each class are returned, and the class with maximal probability else.
threshold	Value replacing cells with probabilities within eps range.
eps	double for specifying an epsilon-range to apply laplace smoothing (to replace zero or close-zero probabilities by theshold.)

An object of class "naiveBayes" including components:

**apriori** Class distribution for the dependent variable.

**tables** A list of tables, one for each predictor variable. For each categorical variable a table giving, for each attribute level, the conditional probabilities given the target class. For each numeric variable, a table giving, for each target class, mean and standard deviation of the (sub-)variable.

## Тема 10. Дерева рішень та ансамблеві методи

Сфера застосування дерев рішень досить широка, в цьому розділі будемо використовувати цей апарат для вирішення завдань класифікації. Величезна перевага дерев рішень в тому, що вони легко інтерпретуються і зрозумілі для людини. Ще однією перевагою дерев рішень є їхня здатність виявляти нетипові випадки (на відміну, наприклад, від логістичної регресії).

Для побудови дерева рішень навчальна вибірка готовиться стандартно, вихідне поле для дерева рішень єдине та дискретне.

Для оцінки якості моделі використовується матриця спряженості.

Ансамблі – поєднання кількох алгоритмів, які навчаються одночасно та виправляють помилки один одного.

Серед ансамблевих методів розглянемо три:

- баггінг (bagging, bootstrap aggregating)
- бустінг (boosting)
- стекінг

Ідея баггінга полягає в тому, що при відсутності великої навчальної вибірки можна створювати багато випадкових вибірок з вихідної простим вибором з заміщенням. Хоча елементи в вибірках можуть перетинатися або дублюватися, на практиці результати об'єднання з багатьох вибірок виявляються точнішими. Метод об'єднує результати передбачення різних класифікаторів, які навчено на випадкових підмножинах. Баггінг є корисним, коли малі зміни в початковій вибірці призводять до суттєвих змін класифікації.

Бустінг – це процедура послідовної композиції алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки композиції попередніх алгоритмів. Тут також робляться вибірки даних, проте вже не за випадковою ознакою. Тепер кожна наступна вибірка складається з тих даних, на яких помилився попередній алгоритм.

Бустінг над вирішальними деревами вважається одним з найбільш ефективних методів з точки зору якості класифікації. У багатьох експериментах спостерігалося практично необмежене зменшення частоти помилок на незалежній тестовій вибірці в міру нарощування композиції. Більш того, якість на тестовій вибірці часто продовжує поліпшуватися навіть після досягнення безпомилкового розпізнавання всієї навчальної вибірки.

Стекінг – ще один спосіб об'єднання класифікаторів, що вводить поняття мета-алгоритму навчання. На відміну від беггінга та бустінга, тут використовуються класифікатори різної природи. Алгоритм стекінгу наступний:

- 1) розбити навчальну вибірку на дві підмножини, що не перетинаються
- 2) навчити кілька базових класифікаторів на першій підмножині
- 3) протестувати базові класифікатори на другій підмножині
- 4) використовуючи розрахункові результати з попереднього пункту як вхідні дані, а справжні класи об'єктів як вихід, навчити мета-алгоритм.

## Лабораторна робота 9

### Classification Tree

#### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
f <- f[-1]
```

Висновок: для побудови моделі банківського скрингу використані дані про наявність прострочених платежів за кредитом.

#### Splitting the dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f$delays, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

Висновок: датасет розподілено на навчальну та тестову вибірки.

#### Fitting

```
# install.packages('rpart')
library(rpart)
class_dt = rpart(delays ~ ., data = f_train)
```

Висновок: базову модель дерева побудовано на основі всіх змінних.

#### Predicting

```
y <- predict(class_dt, f_test[-9], type = 'class')
```

Висновок: визначені класи об'єктів (вектор y).

#### Confusion Matrix

```
cm = table(f_test[, 'delays'], y)
print(cm)

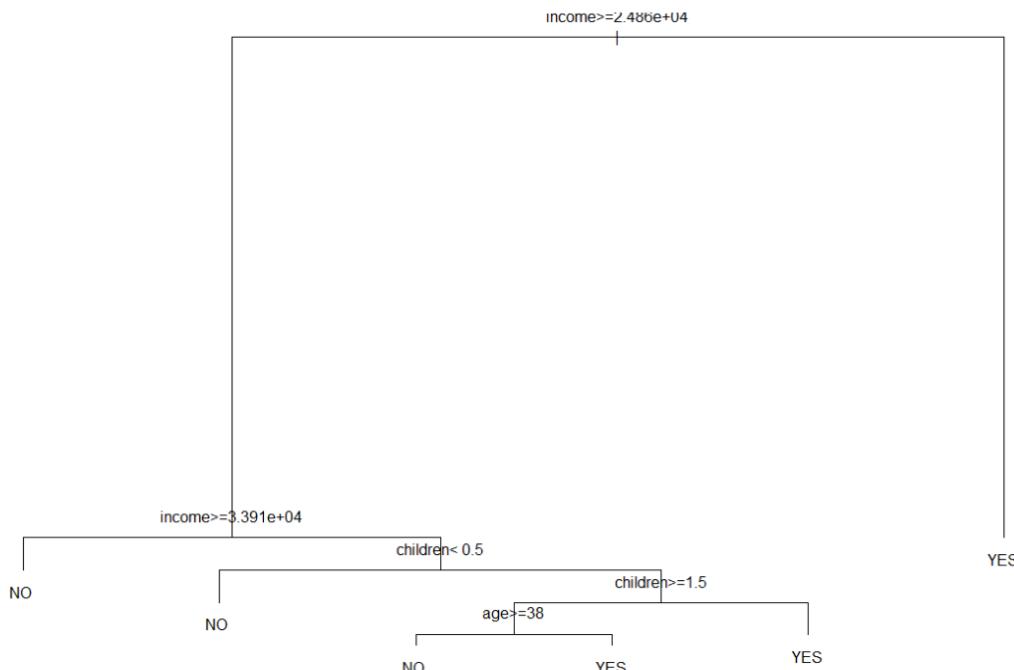
##      y
##      NO YES
##    NO  78  20
##    YES  8  94
```

Кононова К. Ю.

**Висновок:** точність моделі –  $(78+94) / 200 = 86\%$ , частка невірно класифікованих випадків –  $(20+8) / 200 = 14\%$ . Чутливість –  $94 / (8+94) = 92\%$ , специфічність –  $78 / (78+20) = 80\%$ , тобто модель суттєво більш чутлива до виявлення позитивних випадків. У цьому разі – кредиторів, що мають залогованість.

## Plotting the tree

```
plot(class_dt)  
text(class_dt)
```



**Висновок:** візуалізація дозволяє проаналізувати логіку побудови дерева. Зокрема, кредитори, дохід яких менше за три з половиною тисячі, у яких більше однієї дитини, і які молодше 38 років, ймовірно, затримуватимуть відшкодування кредиту

## Less features

```
f <- f[,c('age', 'income', 'delays')]
```

**Висновок:** залишимо для моделювання лише кількісні змінні.

## Features Scaling

```
sc <- scale(f[-3])  
f$age <- sc[,c('age')]  
f$income <- sc[,c('income')]
```

**Висновок:** проведемо шкалювання кількісних змінних.

## Splitting the scaled dataset into the TRAIN set and TEST set

```
set.seed(123)  
library(caTools)  
split = sample.split(f$delays, SplitRatio = 2/3)
```

```
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

**Висновок:** підготований датасет розподілено на навчальну та тестову вибірки.

## Fitting

```
class_ct = rpart(delays ~ ., data = f_train)
```

**Висновок:** проведено навчання моделі дерева рішень.

## Predicting

```
y <- predict(class_ct, f_test[, c('age', 'income')], type = 'class')
```

**Висновок:** визначено класи об'єктів (вектор  $y$ ). Для цього використано параметр  $type = 'class'$ .

## Confusion Matrix

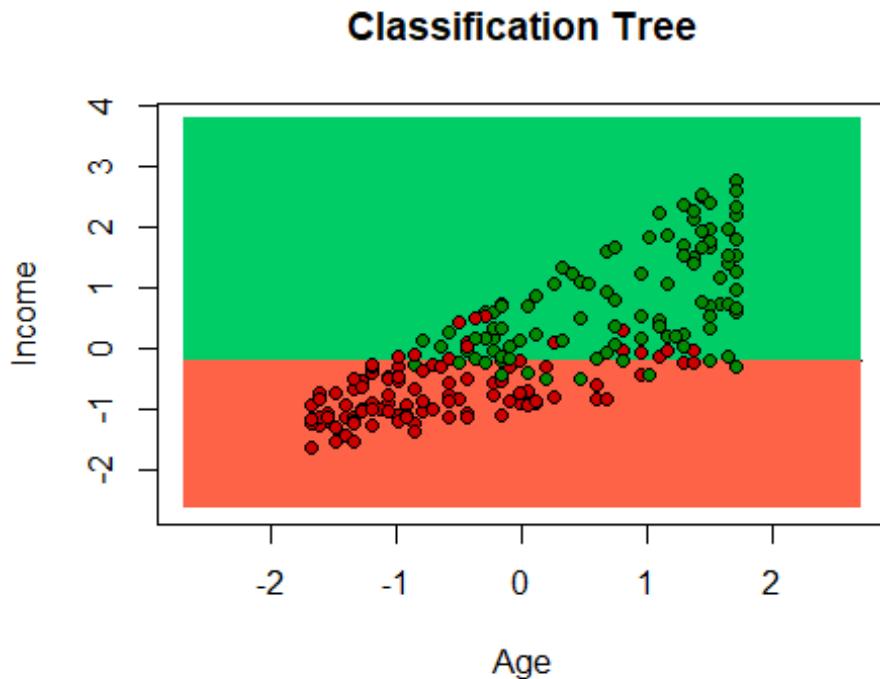
```
cm = table(f_test[, 'delays'], y)
print(cm)

##           y
##      NO YES
##    NO  86  12
##    YES 15  87
```

**Висновок:** точність моделі –  $(86+87) / 200 = 86,5\%$ , частка невірно класифікованих випадків –  $(12+15) / 200 = 13,5\%$ . Чутливість –  $87 / (15+87) = 85\%$ , специфічність –  $86 / (86+12) = 88\%$ , тобто модель більш чутлива до виявлення негативних випадків. У цьому разі – кредиторів, що не мають залогованості.

## Visualising the Test set results

```
#install.packages("ElemStatLearn")
library(ElemStatLearn)
set = f_test[,c('age', 'income', 'delays')]
X1 = seq(min(set['age']) - 1, max(set['age']) + 1, by = 0.01)
X2 = seq(min(set['income']) - 1, max(set['income']) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('age', 'income')
y_grid = predict(class_ct, grid_set, type = 'class')
plot(set[, -3],
     main = 'Classification Tree',
     xlab = 'Age', ylab = 'Income',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 'YES', 'tomato', 'springgreen3'))
points(set, pch = 21, bg = ifelse(set[, 3] == 'YES', 'red3', 'green4'))
```



Висновок: на графіку червоним позначені випадки затримки з поверненням кредиту, зеленим – хороші кредитори. Червоним виділена зона високої ймовірності неповернення кредиту. Модель описує нелінійний варіант розподіляючої кривої.

### Fitting Random Forest Classification to the Training set

```
# install.packages('randomForest')
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

set.seed(123)
class_rf = randomForest(delays ~ ., data = f_train, ntree = 50)
```

Висновок: проведено навчання моделі випадкового лісу.

### Predicting

```
y <- predict(class_rf, f_test[, c('age', 'income')], type = 'class')
```

Висновок: визначені класи об'єктів (вектор  $y$ ). Для цього використано параметр  $type = 'class'$ .

### Confusion Matrix

```
cm = table(f_test[, 'delays'], y)
print(cm)

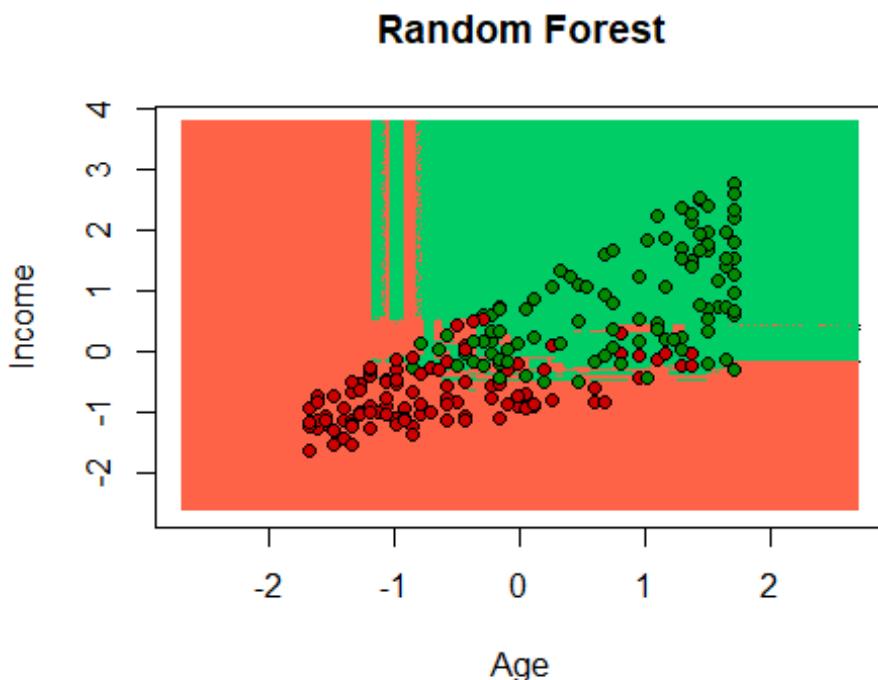
##      y
##      NO YES
```

```
##   NO  86  12
##   YES 15  87
```

**Висновок:** точність моделі –  $(86+87) / 200 = 86,5\%$ , частка невірно класифікованих випадків –  $(12+15) / 200 = 13,5\%$ . Чутливість –  $87 / (15+87) = 85\%$ , специфічність –  $86 / (86+12) = 88\%$ , тобто модель показує тіж самі характеристики, що й для дерева рішень, вона більш чутлива до виявлення негативних випадків. У цьому разі – кредиторів, що не мають залогованості.

## Visualising the Test set results

```
set = f_test[,c('age', 'income', 'delays')]
X1 = seq(min(set['age']) - 1, max(set['age']) + 1, by = 0.01)
X2 = seq(min(set['income']) - 1, max(set['income']) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('age', 'income')
y_grid = predict(class_rf, grid_set, type = 'class')
plot(set[, -3],
      main = 'Random Forest',
      xlab = 'Age', ylab = 'Income',
      xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 'YES', 'tomato', 'springgreen3'))
points(set, pch = 21, bg = ifelse(set[, 3] == 'YES', 'red3', 'green4'))
```



**Висновок:** на графіку червоним позначені випадки затримки з поверненням кредиту, зеленим – хороші кредитори. Червоним виділена зона високої ймовірності неповернення кредиту. Модель описує нелінійний варіант розподіляючої кривої.

## Fitting XGBoost to the Training set

```
# install.packages('xgboost')
library(xgboost)

## Warning: package 'xgboost' was built under R version 3.6.2

class_xboost = xgboost(data = as.matrix(f_train[-9]), label = f_train$delays, nrounds = 10)

## [1] train-rmse:0.382970
## [2] train-rmse:0.294708
## [3] train-rmse:0.234244
## [4] train-rmse:0.190218
## [5] train-rmse:0.160858
## [6] train-rmse:0.138820
## [7] train-rmse:0.125096
## [8] train-rmse:0.115156
## [9] train-rmse:0.106145
## [10] train-rmse:0.101461
```

**Висновок:** проведено навчання моделі XGBoost протягом 10 раундів.

## Predicting the Test set results

```
y_pred = predict(class_xboost, newdata = as.matrix(f_test[-9]))
y_pred = (y_pred >= 0.5)
```

## Making the Confusion Matrix

```
cm = table(f_test[, 9], y_pred)
print(cm)

##      y_pred
##      FALSE TRUE
##      0     82    6
##      1     16   75
```

**Висновок:** точність моделі –  $(82+75) / 179 = 87,7\%$ , частка невірно класифікованих випадків –  $(16+6) / 179 = 12,3\%$ . Чутливість –  $75 / (16+75) = 82\%$ , специфічність –  $82 / (82+6) = 93\%$ .

## Applying k-Fold Cross Validation

```
# install.packages('caret')
library(caret)

folds = createFolds(f_train$delays, k = 10)
cv = lapply(folds, function(x) {
  train_fold = f_train[-x, ]
  test_fold = f_train[x, ]
```

```

class_xboost = xgboost(data = as.matrix(f_train[-9]), label = f_train$de
lays, nrounds = 10)
y_pred = predict(class_xboost, newdata = as.matrix(test_fold[-9]))
y_pred = (y_pred >= 0.5)
cm = table(test_fold[, 9], y_pred)
accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
return(accuracy)
}

accuracy = mean(as.numeric(cv))
print(accuracy)

## [1] 1

```

## Довідка: randomForest

```

## S3 method for class 'formula'
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
## Default S3 method:
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
               max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
## S3 method for class 'randomForest'
print(x, ...)

```

### Arguments

data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which randomForest is called from.
subset	an index vector indicating which rows should be used. (NOTE: If given, this argument must be named).
na.action	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named).
x, formula	a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the print method, an randomForest object).
y	A response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, randomForest will run in unsupervised mode.
xtest	a data frame or matrix (like x) containing predictors for the test set.
ytest	response for the test set.
ntree	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.

mtry	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification ( $\sqrt{p}$ ) where $p$ is number of variables in $x$ ) and regression ( $p / 3$ ).
replace	Should sampling of cases be done with or without replacement?
classwt	Priors of the classes. Need not add up to one. Ignored for regression.
cutoff	(Classification only) A vector of length equal to number of classes. The ‘winning’ class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is $1/k$ where $k$ is the number of classes (i.e., majority vote wins).
strata	A (factor) variable that is used for stratified sampling.
sampsize	Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
nodesize	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
maxnodes	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by nodesize). If set larger than maximum possible, a warning is issued.
importance	Should importance of predictors be assessed?
localImp	Should casewise importance measure be computed? (Setting this to TRUE will override importance).
nPerm	Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.
proximity	Should proximity measure among the rows be calculated?
oob.prox	Should proximity be calculated only on “out-of-bag” data?
norm.votes	If TRUE (default), the final result of votes are expressed as fractions. If FALSE, raw vote counts are returned (useful for combining results from different runs). Ignored for regression.
do.trace	If set to TRUE, give a more verbose output as randomForest is run. If set to some integer, then running output is printed for every do.tracetrees.
keep.forest	If set to FALSE, the forest will not be retained in the output object. If xtest is given, defaults to FALSE.
corr.bias	perform bias correction for regression? Note: Experimental. Use at your own risk.
keep.inbag	Should an n by ntree matrix be returned that keeps track of which samples are “in-bag” in which trees (but not how many times, if sampling with replacement)

... optional parameters to be passed to the low level function randomForest.default.

An object of class `randomForest`, which is a list with the following components:

<code>call</code>	the original call to <code>randomForest</code>
<code>type</code>	one of regression, classification, or unsupervised.
<code>predicted</code>	the predicted values of the input data based on out-of-bag samples.
<code>importance</code>	a matrix with $n_{\text{class}} + 2$ (for classification) or two (for regression) columns. For classification, the first $n_{\text{class}}$ columns are the class-specific measures computed as mean decrease in accuracy. The $n_{\text{class}} + 1$ st column is the mean decrease in accuracy over all classes. The last column is the mean decrease in Gini index. For Regression, the first column is the mean decrease in accuracy and the second the mean decrease in MSE. If <code>importance = FALSE</code> , the last measure is still returned as a vector.
<code>importanceSD</code>	The “standard errors” of the permutation-based importance measure. For classification, a $p$ by $n_{\text{class}} + 1$ matrix corresponding to the first $n_{\text{class}} + 1$ columns of the importance matrix. For regression, a length $p$ vector.
<code>localImp</code>	a $p$ by $n$ matrix containing the casewise importance measures, the $[i, j]$ element of which is the importance of $i$ -th variable on the $j$ -th case. <code>NULL</code> if <code>localImp = FALSE</code> .
<code>ntree</code>	number of trees grown.
<code>mtry</code>	number of predictors sampled for splitting at each node.
<code>forest</code>	(a list that contains the entire forest; <code>NULL</code> if <code>randomForest</code> is run in unsupervised mode or if <code>keep.forest = FALSE</code> ).
<code>err.rate</code>	(classification only) vector error rates of the prediction on the input data, the $i$ -th element being the (OOB) error rate for all trees up to the $i$ -th.
<code>confusion</code>	(classification only) the confusion matrix of the prediction (based on OOB data).
<code>votes</code>	(classification only) a matrix with one row for each input data point and one column for each class, giving the fraction or number of (OOB) ‘votes’ from the random forest.
<code>oob.times</code>	number of times cases are ‘out-of-bag’ (and thus used in computing OOB error estimate)
<code>proximity</code>	if <code>proximity = TRUE</code> when <code>randomForest</code> is called, a matrix of proximity measures among the input (based on the frequency that pairs of data points are in the same terminal nodes).
<code>mse</code>	(regression only) vector of mean square errors: sum of squared residuals divided by $n$ .
<code>rsq</code>	(regression only) “pseudo R-squared”: $1 - \text{mse} / \text{Var}(y)$ .

`test` if test set is given (through the `xtest` or additionally `ytestarguments`), this component is a list which contains the corresponding predicted, err.rate, confusion, votes (for classification) or predicted, mse and rsq (for regression) for the test set.  
 If `proximity` = TRUE, there is also a component, `proximity`, which contains the proximity among the test set as well as proximity between test and training data.

### Переваги і недоліки розглянутих моделей

Модель	Переваги	Недоліки
Логістична регресія	Ймовірнісний підхід, що дозволяє оцінити значущість факторів	Припущення лінійної розподільності
Метод опорних векторів	Не є чутливим до перенавчання	Необхідно розрізняти лінійний та нелінійний випадки
Метод K найближчих сусідів	Простий, швидкий і ефективний	Необхідність вручну підбирати кількість сусідів
Байесівський класифікатор	Стійкий до викидів ймовірнісний підхід, що працює в лінійно нерозподільних просторах	Припущення про рівну статистичну значущість факторів
Дерева рішень	Добре інтерпретуються, не потребують шкалювання, дозволяють моделювати лінійні та нелінійні залежності	Погано працюють на малих вибірках, велика ймовірність перенавчання
Випадковий ліс	Висока точність, особливо під час моделювання нелінійних залежностей	Не інтерпретуються, легко перенавчати, потрібно вручну підбирати кількість дерев

### **Питання для самоперевірки**

1. Перелічіть види моделей класифікації.
2. Особливості моделі логістичної регресії.
3. Переваги та недоліки моделі логістичної регресії.
4. У чому особливості методу опорних векторів? Переваги і недоліки моделі.
5. У чому полягає метод К найближчих сусідів? Зазначте його переваги і недоліки.
6. Яка основна ідея байєсівського класифікатора? Переваги і недоліки моделі.
7. Особливості використання дерева рішень в задачах класифікації.
8. Переваги та недоліки дерева рішень в задачах класифікації.
9. Переваги і недоліки моделі випадкового лісу в задачах класифікації.
10. Як будується і для чого використовується таблиця спряженості.
11. ROC-аналіз для оцінки якості моделей класифікації.

### **Самостійна робота 3**

Зібрати дані, що підлягають класифікації (наприклад, про рівень соціально-економічного розвитку країн світу). На їх основі побудувати моделі класифікації, провести їх порівняльний аналіз і зробити висновки про доцільність застосування.

## РОЗДІЛ 4. КЛАСТЕРИЗАЦІЯ

### Тема 11. Ієрархічна кластеризація

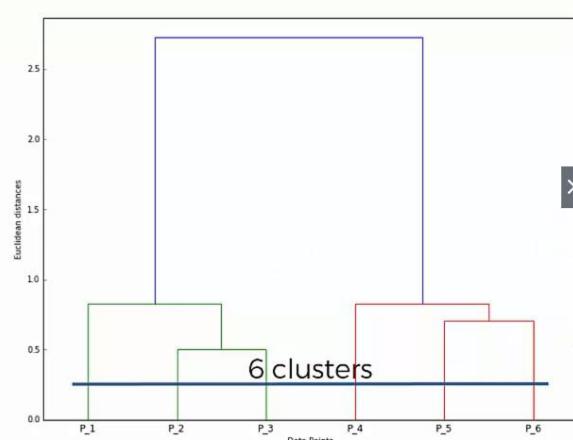
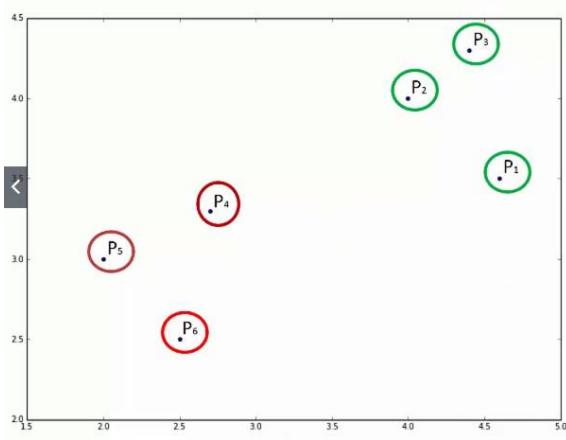
Ієрархічна кластеризація – метод кластерного аналізу, що спрямований на створення ієрархії кластерів. Стратегії ієрархічної кластеризації поділяються на два типи:

1) дивізійна – «згори донизу»: всі спостереження розміщаються в одному кластері, розщеплення виконуються, рекурсивно пересуваючись вниз по ієрархії;

2) агломеративна – «знизу догори»: кожне спостереження розміщується у власному кластері, пари кластерів об'єднуються, пересуваючись вгору по ієрархії.

Алгоритм агломеративної кластеризації полягає в такому:

- кожна точка розташовується у своєму кластері;
- попарні відстані між центрами кластерів упорядковуються за зростанням;
- пара найближчих кластерів об'єднується в один і перераховується центр кластера;
- процедура повторюється доти, доки всі дані не об'єднаються в один кластер.

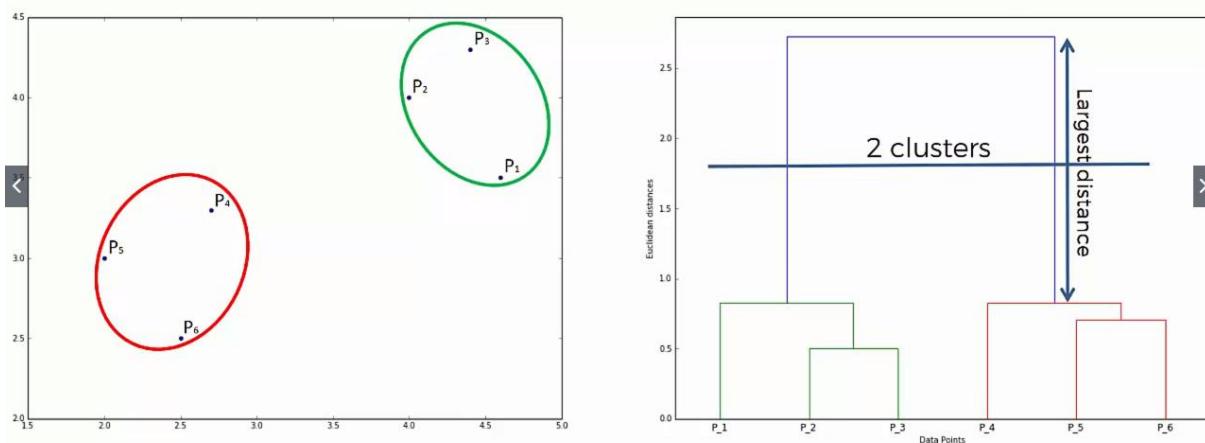


Процес пошуку найближчих кластерів може відбуватися з використанням різних методів об'єднання точок з двох кластерів:

- single linkage – мінімум попарних відстаней між точками;
- complete linkage – максимум попарних відстаней між точками;
- average linkage – середнє значення попарних відстаней між точками;
- centroid linkage – відстань між центроїдами двох кластерів.

Перевага перших трьох підходів порівняно з четвертим полягає в тому, що для них не потрібно перераховувати відстані кожен раз після об'єднання, що сильно знижує обчислювальну складність алгоритму.

За підсумками виконання алгоритму будується дендрограмма. На її основі можна визначити, на якому етапі найкраще зупинити алгоритм.



## Лабораторна робота 10

### Hierarchical clustering

#### Download the data

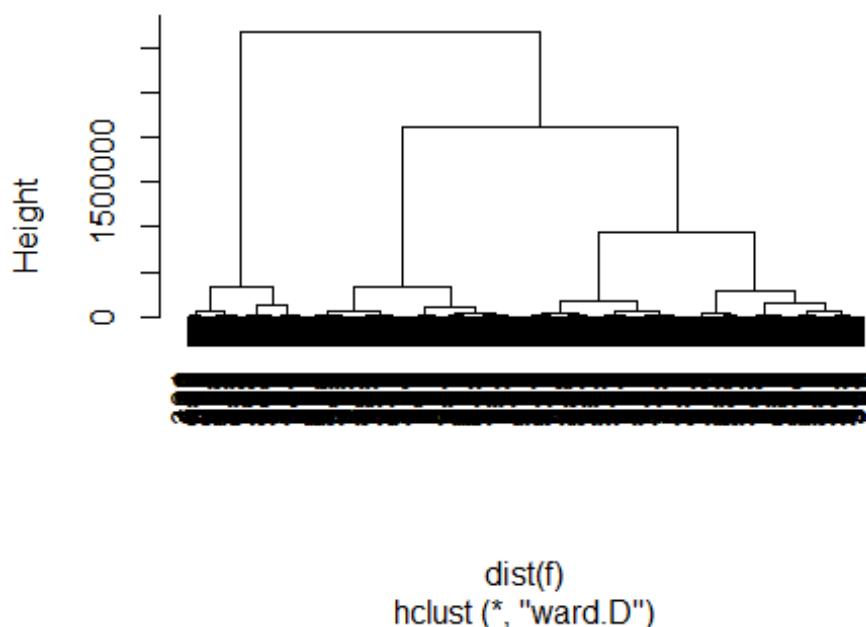
```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
f <- f[, c('age', 'income')]
```

Висновок: для побудови моделі кластеризації використано дані про наявність прострочених платежів по кредиту. Залишмо для моделювання лише кількісні змінні. Датасет не потребує розподілу на навчальну та тестову вибірки, не потребує шкалювання.

#### Hierarchical clustering

```
model_hc <- hclust(dist(f), method = "ward.D" )
plot(model_hc, main = paste('Dendrogram'))
```

### Dendrogram



dist(f)  
hclust (\*, "ward.D")

Висновок: на основі навчальної вибірки побудовано дендрограму з використанням методу Ward.D.

### Fitting HC to the dataset

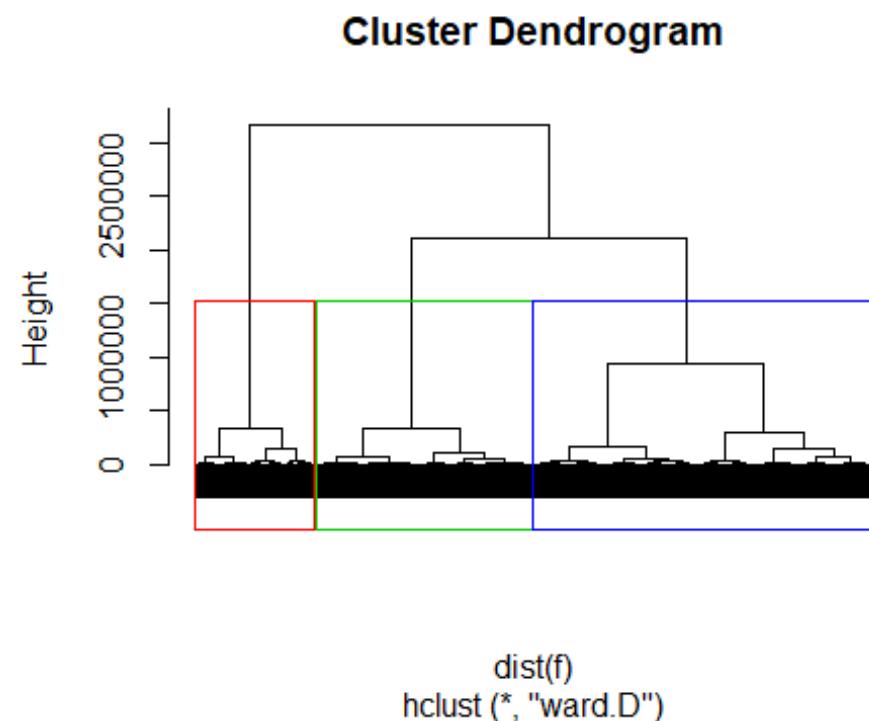
```
y_hc <- cutree(model_hc, k = 3)
#cluster cores
aggregate(f, by=list(y_hc), FUN=mean)

##   Group.1      age    income
## 1      1 29.45596 14284.70
## 2      2 44.82392 28358.90
## 3      3 59.05660 49258.91
```

Висновок: на основі аналізу дендрограми виявлено три кластера: 1 – “молоді та бідні”, 2 – “середнього віку та рівня доходів”, 3 – “дорослі та багаті”. Розраховано характеристики типового об’єкту кластерів.

### Plotting the dendrogram

```
plot(model_hc, cex = 0.7, labels = FALSE)
rect.hclust(model_hc, k = 3, border = 2:5)
```

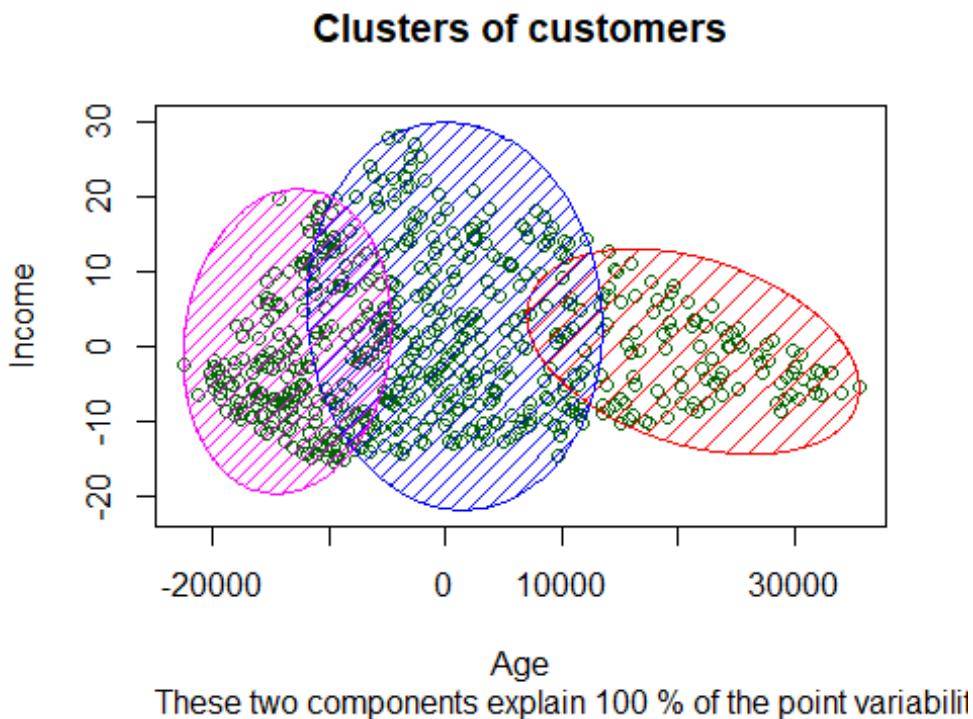


Висновок: проведено візуалізацію кластерів на дендрограмі.

### Visualising the clusters

```
library(cluster)
clusplot(f,
         y_hc,
         lines = 0,
         shade = TRUE,
```

```
color = TRUE,  
labels = 0,  
plotchar = FALSE,  
span = TRUE,  
main = paste('Clusters of customers'),  
xlab = 'Age',  
ylab = 'Income')
```



**Висновок:** проведено візуалізацію кластерів на датасеті.

### Довідка: hclust

```
hclust(d, method = "complete", members = NULL)  
  
## S3 method for class 'hclust'  
plot(x, labels = NULL, hang = 0.1, check = TRUE,  
      axes = TRUE, frame.plot = FALSE, ann = TRUE,  
      main = "Cluster Dendrogram",  
      sub = NULL, xlab = NULL, ylab = "Height", ...)
```

#### Arguments

- d a dissimilarity structure as produced by dist.
- method the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

members	NULL or a vector with length size of d. See the ‘Details’ section.
x	an object of the type produced by hclust.
hang	The fraction of the plot height by which labels should hang below the rest of the plot. A negative value will cause the labels to hang down from 0.
check	logical indicating if the x object should be checked for validity. This check is not necessary when x is known to be valid such as when it is the direct result of hclust(). The default is check = TRUE, as invalid inputs may crash R due to memory violation in the internal C plotting code.
labels	A character vector of labels for the leaves of the tree. By default the row names or row numbers of the original data are used. If labels = FALSE no labels at all are plotted.
axes, frame.plot, ann	logical flags as in <a href="#">plot.default</a> .
main, sub, xlab, ylab	character strings for <a href="#">title</a> . sub and xlab have a non-NULL default when there’s a tree\$call.
...	Further graphical arguments. E.g., cex controls the size of the labels (if plotted) in the same way as <a href="#">text</a> .

An object of class **hclust** which describes the tree produced by the clustering process. The object is a list with components:

merge	an $n-1$ by 2 matrix. Row $i$ of merge describes the merging of clusters at step $i$ of the clustering. If an element $j$ in the row is negative, then observation $-j$ was merged at this stage. If $j$ is positive then the merge was with the cluster formed at the (earlier) stage $j$ of the algorithm. Thus negative entries in merge indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.
height	a set of $n-1$ real values (non-decreasing for ultrametric trees). The clustering <i>height</i> : that is, the value of the criterion associated with the clustering method for the particular agglomeration.
order	a vector giving the permutation of the original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix merge will not have crossings of the branches.
labels	labels for each of the objects being clustered.
call	the call which produced the result.
method	the cluster method that has been used.
dist.method	the distance that has been used to create d (only returned if the distance object has a "method" attribute).

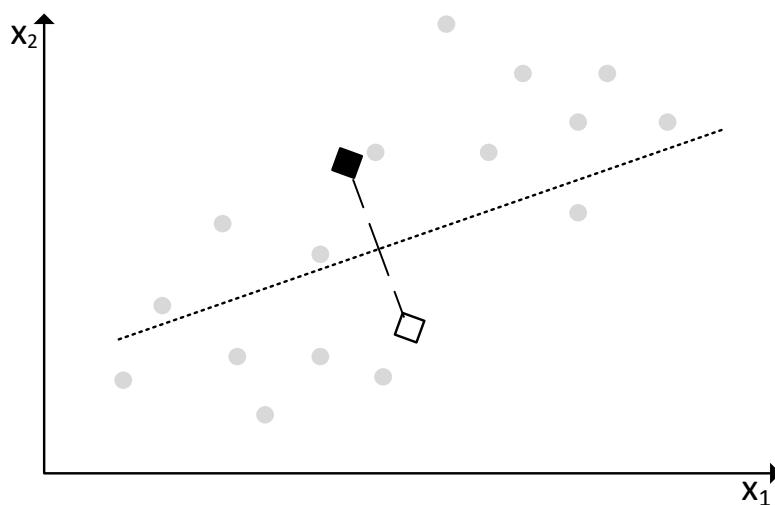
## Тема 12. Кластеризація на основі k-means

Основою роботи алгоритму  $k$ -середніх ( $k$ -means) є принцип оптимального в певному сенсі розбиття множини даних на  $k$  кластерів. Алгоритм намагається згрупувати дані в кластери так, щоб цільова функція алгоритму розбиття досягала екстремуму.

Нехай об'єкти, що підлягають кластеризації, описуються вектором параметрів  $X^p$ . Введемо множину кластерів і визначимо їх ядра  $C^k$  як типові об'єкти свого кластера. Для оцінки близькості об'єкта до ядра використовується евклідова міра близькості  $D(x^p, c^k)$ , що тим менше, чим більше об'єкт схожий на ядро класу, тобто під час розбиття на кластери повинна бути мінімізована сумарна міра близькості для всієї множини вхідних об'єктів

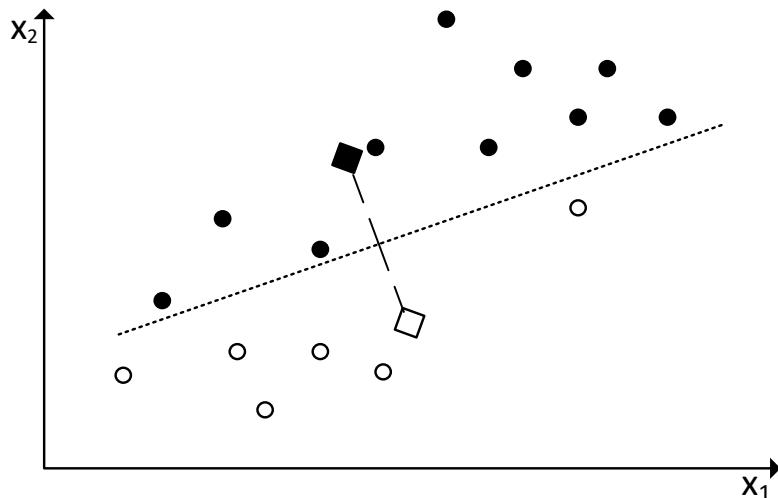
$$D(x^p, c^k) = \sum_p \sum_i (x_i^p - c_i^k)^2 \rightarrow \min$$

Розглянемо загальний алгоритм з фіксованою кількістю ядер  $k$  (кількість ядер вибирається заздалегідь з огляду на конкретне завдання, початкові значення ядер можуть вибиратися випадковими, однаковими або за іншими евристичними правилами).

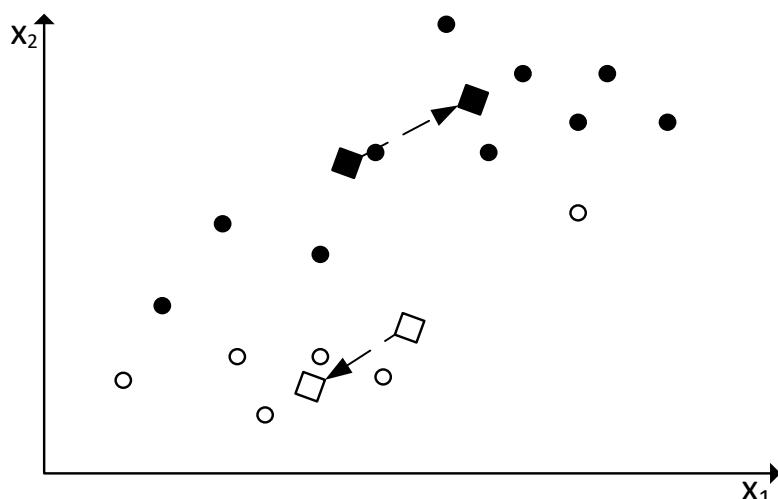


Кожна ітерація алгоритму складається з двох етапів:

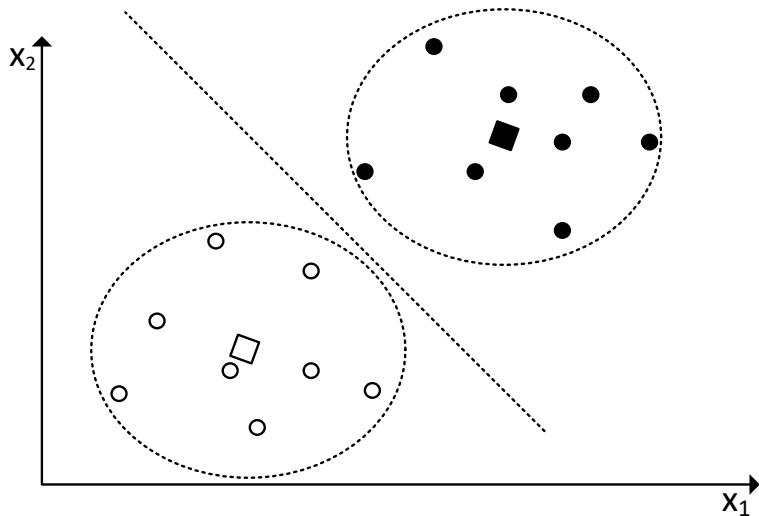
1. За незмінних ядер шукається таке розбиття об'єктів на класи, щоб мінімізувати сумарну міру близькості між об'єктами та ядрами класів:  $\min \{ \sum D(x^p, c^k) \}$ . Результат етапу – створення функції, що розбиває об'єкти на класи.



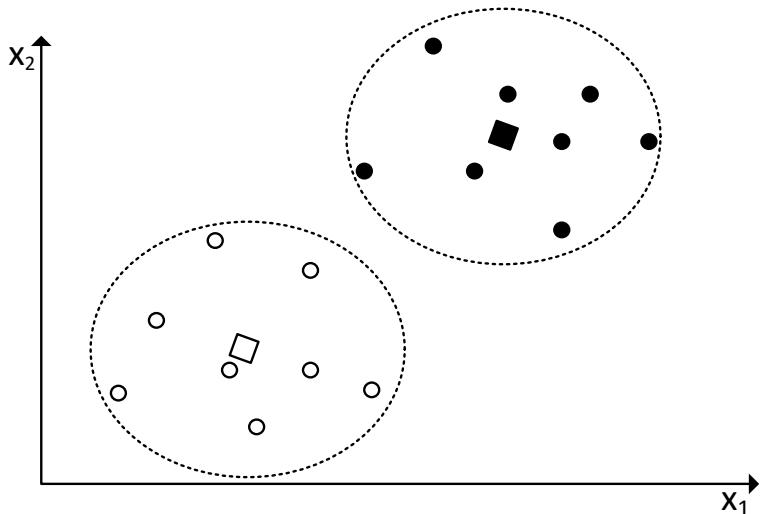
2. За незмінного розбиття ядра класів налаштовуються так, щоб в межах кожного класу сумарна міра близькості ядра цього класу та об'єктів, що йому належать, була мінімальною. Результат цього етапу – нова сукупність ядер.



Ітерації повторюються доти доки розбиття не стабілізується.



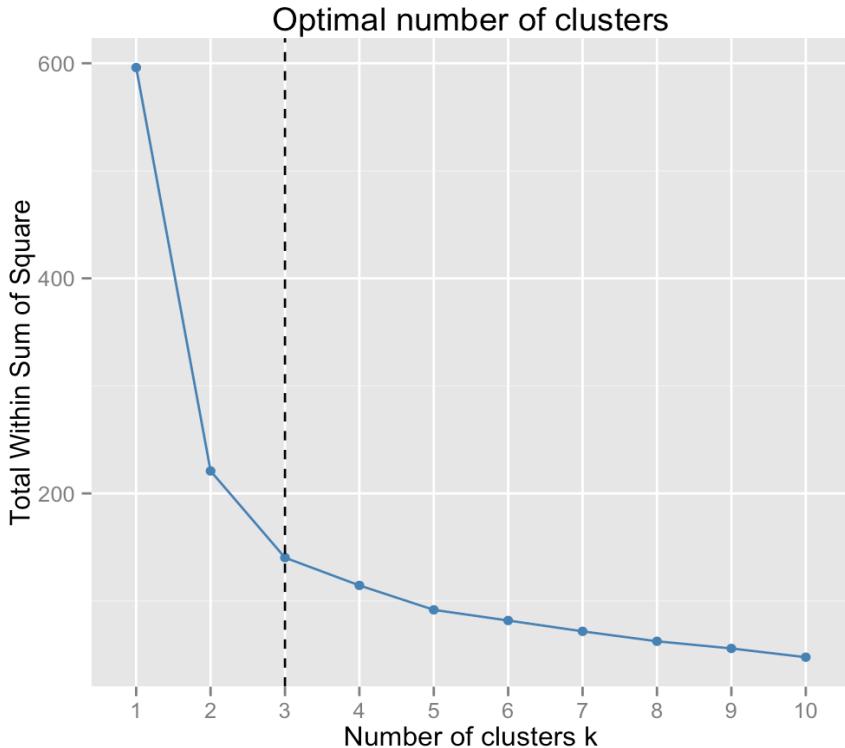
Для вибору кількості кластерів на кожному етапі розбиття, послідовно збільшуючи кількість кластерів, для кожного кластера оцінюється сумарна відстань від ядра до об'єктів кластера. Потім знаходиться сумарна міра щільності кластерів.



$$WCSS = \sum_{cluster_i} \sum_{object_j} dist(c_i, o_j)$$

Сумарна міра щільності кластерів зменшується зі зростанням кількості кластерів. Точка, де швидкість падіння зменшується максимально, вважається оптимальною кількістю кластерів (правило ліктя)

$$\frac{WCSS_t - WCSS_{t+1}}{WCSS_{t-1} - WCSS_t} \rightarrow \min$$



Якщо кількість кластерів визначити важко, то можна використовувати алгоритм *g-means*. Він визначає кількість кластерів в моделі на підставі послідовного виконання статистичного тесту на те, що дані всередині кластера підкоряються гауссівському (Gaussian, звідси назва алгоритму) закону розподілу. Якщо тест дає негативний результат, кластер розбивається на два нових кластера з центрами, розташованими на осі головних компонент.

Важливо пам'ятати, що алгоритми *k-means* та *g-means* орієнтовані на гіпотезу про компактності, яка передбачає, що дані навчальної вибірки утворюють компактні області. В іншому разі кластери, знайдені цими алгоритмами, будуть малоінформативними.

## Лабораторна робота 11

### K-Means

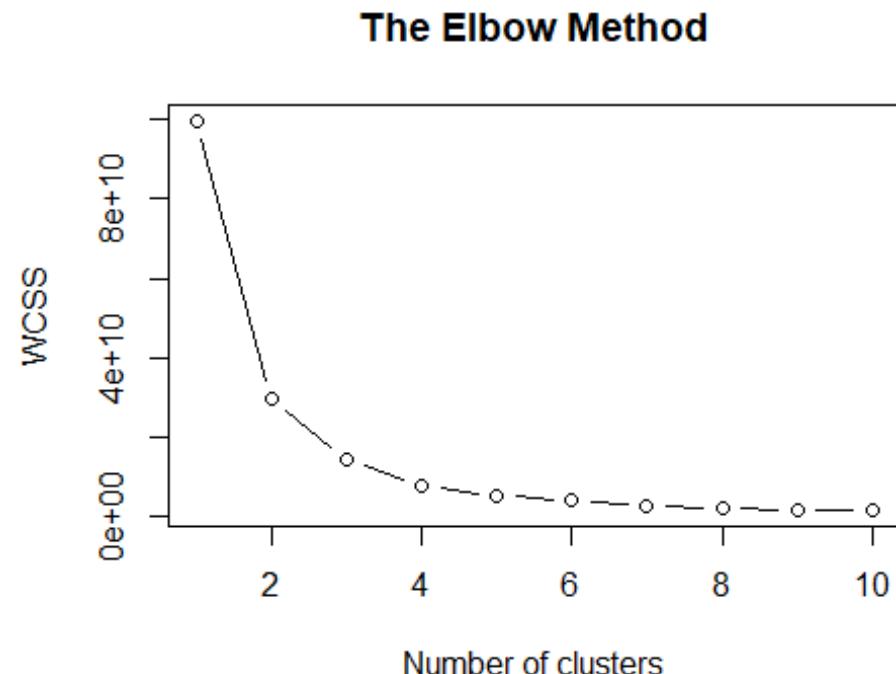
#### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
f <- f[, c('age', 'income')]
```

Висновок: для побудови моделі кластеризації використано дані про наявність прострочених платежів по кредиту. Залишмо для моделювання лише кількісні змінні. Датасет не потребує розподілу на навчальну та тестову вибірки, не потребує шкалювання.

#### Elbow method to find optimal number of clusters

```
library(stats)
wcss = vector()
for (i in 1:10) wcss[i] = sum(kmeans(f, i)$withinss)
plot(1:10,
      wcss,
      type = 'b',
      main = paste('The Elbow Method'),
      xlab = 'Number of clusters',
      ylab = 'WCSS')
```



**Висновок:** з використанням метода ліктя зроблено висновок про доцільність виявлення трьох кластерів.

## Fitting K-Means to the dataset

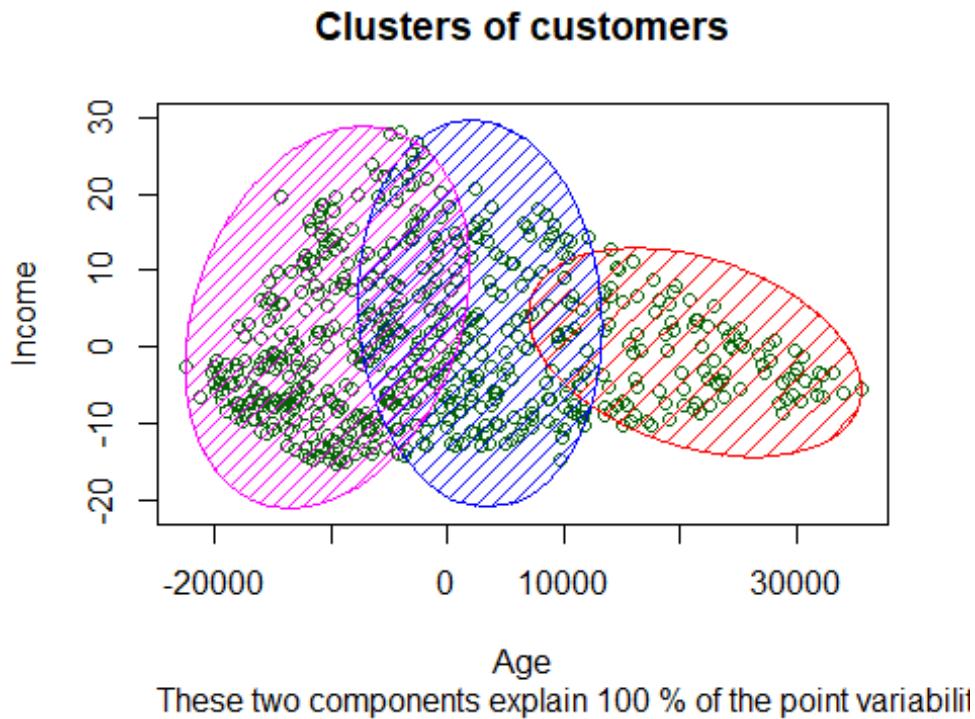
```
set.seed(29)
model_km = kmeans(f, 3)
#cluster cores
y_km = model_km$cluster
aggregate(f, by=list(y_km), FUN=mean)

##   Group.1      age    income
## 1      1 32.36996 16487.89
## 2      2 46.78733 30732.04
## 3      3 59.05660 49258.91
```

**Висновок:** на основі методу k-середніх описано три кластера: 1 – “молоді та бідні”, 2 – “середнього віку та рівня доходів”, 3 – “дорослі та багаті”. Розраховано характеристики типового об'єкту кластерів.

## Visualising the clusters

```
library(cluster)
clusplot(f,
         y_km,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels= 0,
         plotchar = FALSE,
         span = TRUE,
         main = paste('Clusters of customers'),
         xlab = 'Age',
         ylab = 'Income')
```



Висновок: проведено візуалізацію кластерів на датасеті.

## Comparing to HC

```
library(clusteval)
cluster_similarity(y_hc,y_km)

##[1] 0.6088943
```

Висновок: отримані кластери перетинаються на 60,8%

## Довідка: kmeans

```
kmeans(x, centers, iter.max = 10, nstart = 1,
        algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
## S3 method for class 'kmeans'
fitted(object, method = c("centers", "classes"), ...)
```

### Arguments

- x numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
- centers either the number of clusters, say  $k$ , or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in x is chosen as the initial centres.
- iter.max the maximum number of iterations allowed.

nstart	if centers is a number, how many random sets should be chosen?
algorithm	character: may be abbreviated. Note that "Lloyd" and "Forgy" are alternative names for one algorithm.
object	an <b>R</b> object of class "kmeans", typically the result ob of ob <- kmeans(..).
method	character: may be abbreviated. "centers" causes fitted to return cluster centers (one for each input point) and "classes" causes fitted to return a vector of class assignments.
trace	logical or integer number, currently only used in the default method ("Hartigan-Wong"): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.
...	not used.

kmeans returns an object of class "kmeans" which has a print and a fitted method. It is a list with at least the following components:

cluster	A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
centers	A matrix of cluster centres.
totss	The total sum of squares.
withinss	Vector of within-cluster sum of squares, one component per cluster.
tot.withinss	Total within-cluster sum of squares, i.e. sum(withinss).
betweenss	The between-cluster sum of squares, i.e. totss-tot.withinss.
size	The number of points in each cluster.
iter	The number of (outer) iterations.
ifault	integer: indicator of a possible algorithm problem – for experts.

## Якість кластеризації

Після кластеризації виникає питання, наскільки воно стійке та статистично значуще. Існує емпіричне правило – стійка група повинна зберігатися зі зміною методів кластеризації: наприклад, якщо результати ієрархічного кластерного аналізу мають частку збігів більше за 70 % з угруппуванням за методом k-середніх, то береться припущення про стійкість.

У теоретичному плані проблема перевірки адекватності кластеризації не вирішена, можна виділити кілька підходів до валідації кластерів:

- зовнішня валідація, що полягає в порівнянні підсумків кластерного аналізу із заздалегідь відомим результатом (тобто мітки кластерів відомі априорі);
- відносна валідація, що оцінює структуру кластерів, змінюючи різні параметри одного і того ж алгоритму (наприклад, кількість груп  $k$ );
- внутрішня валідація, що використовує внутрішню інформацію процесу об'єднання в кластери (якщо зовнішньої інформації немає);
- оцінка стабільності об'єднання в кластери (або спеціальна версія внутрішньої валідації), що використовує методи ресемплінга.

Одна з проблем машинного навчання без вчителя полягає в тому, що методи кластеризації формуватимуть групи, навіть якщо аналізована сукупність даних є повністю випадковою структурою. Тому першим завданням валідації, що рекомендується виконати перед початком кластерного аналізу, є оцінка загальної схильності наявних даних до об'єднання в кластери (clustering tendency).

Статистика Хопкінса (Hopkins) є одним з індикаторів тенденції до групування. Для її розрахунку створюється  $B$  псевдо-сукупностей даних, згенерованих випадково на основі розподілу з тим же стандартним відхиленням, що і оригінальна сукупність даних. Для кожного спостереження  $i$  з  $n$  розраховується середня відстань до  $k$  найближчих сусідів:  $w_i$  між реальними об'єктами і  $q_i$  між штучними об'єктами і їх найближчими реальними сусідами.

Тоді статистика Хопкінса

$$H_{ind} = \sum w_i / (\sum q_i + \sum w_i),$$

що перевищує 0,5, буде відповідати нульовій гіпотезі про те, що  $q_i$  і  $w_i$  подібні, а об'єкти, що групуються, розподілені випадково і є однорідними. Величина  $H_{ind} < 0,25$  на 90 %-ому рівні впевненості вказує на наявну тенденцію до групування даних.

### Переваги і недоліки розглянутих моделей

Модель	Переваги	Недоліки
Ієрархічна кластеризація	Оптимальна кількість кластерів може бути отримана безпосередньо з моделі, легко інтерпретується дендрограмма кластерів	Не застосовується для великих вибірок
Метод k-середніх	Простий для розуміння, легко адаптується, добре працює і на великих, і на малих вибірках	Необхідно визначати кількість кластерів

#### Питання для самоперевірки

1. Перелічіть види моделей кластеризації.
2. Особливості моделі ієрархічної кластеризації.
3. Переваги та недоліки моделі ієрархічної кластеризації.
4. Особливості моделі k-середніх.
5. Переваги та недоліки моделі k-середніх.
6. Правило ліктя в задачах кластеризації.

#### Самостійна робота 4

Зібрати дані, що підлягають кластеризації (наприклад, дані з попереднього завдання без меток класів). Побудувати відповідні моделі, провести їх порівняльний аналіз і зробити висновки про доцільність застосування.

## РОЗДІЛ 5. АСОЦІАТИВНІ ПРАВИЛА

### Тема 13. Побудова асоціативних правил

Перший алгоритм пошуку асоціативних правил був розроблений 1993 року співробітниками дослідницького центру IBM. Щороку з'являлося кілька нових алгоритмів (DHP, Partition, DIC тощо), з яких найбільш відомим є алгоритм Apriori.

*Асоціативні правила* являють собою механізм знаходження логічних закономірностей ( $X \rightarrow Y$ ) між пов'язаними елементами (подіями або об'єктами). Вперше ця задача була запропонована для знаходження типових шаблонів покупок, що здійснюються в супермаркетах, тому іноді її називають аналізом ринкового кошика (market basket analysis).

Нехай є база даних, що складається з купівельних транзакцій. Кожна транзакція – це сукупність товарів, куплених покупцем за один візит.

Метою аналізу є встановлення таких залежностей: якщо в транзакції зустрілася деяка сукупність елементів  $X$ , то на підставі цього можна зробити висновок про те, що інша сукупність елементів  $Y$  також повинна з'явитися в цій транзакції. Встановлення таких залежностей дає нам можливість знаходити інтуїтивно зрозумілі правила.

Виділяють три види правил:

- 1) *корисні* правила, що містять інформацію, що раніше була невідома, але має логічне пояснення;
- 2) *тривіальні* правила, що містять легко зрозумілу інформацію, що відображатиме відомі закони в досліджуваній сфері, і тому не має користі;
- 3) *незрозумілі* правила, що містять інформацію, яку неможливо пояснити (такі правила або отримують на основі аномальних вихідних даних, або вони містять глибоко приховані закономірності, і тому для інтерпретації незрозумілих правил потрібен додатковий аналіз).

Пошук асоціативних правил зазвичай виконують в два етапи:

- 1) в пулі наявних ознак знаходять комбінації елементів, що найбільш часто зустрічаються;
- 2) з цих сукупностей формують асоціативні правила.

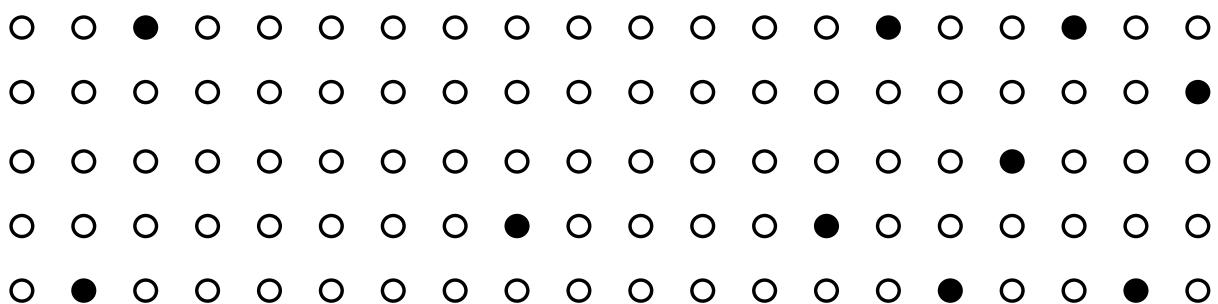
Для оцінки корисності правил використовуються різні частотні критерії. Найважливішими з них є підтримка (support) і достовірність (confidence).

*Підтримкою (s, support)* називають частку (або відсоток) транзакцій, що містять певну сукупність даних.

$S(Y) = \text{кількість транзакцій, що містять } Y / \text{загальна кількість транзакцій}$

Наприклад, 10 % від загальної кількості всіх транзакцій містять сир.

$$\text{Support} = \frac{10}{100} = 10\%$$

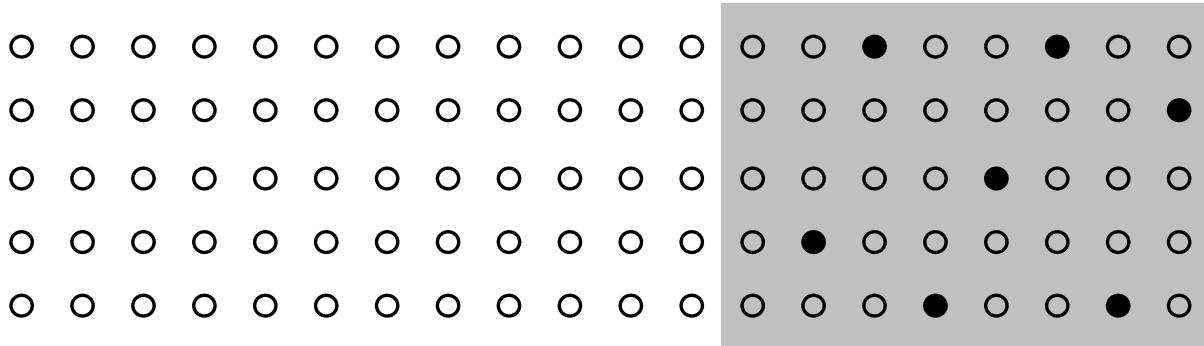


*Достовірність (c, confidence)* – співвідношення кількості транзакцій, що містять всі елементи, які входять до правила, і кількості транзакцій, що містять елементи, які входять до умови. Достовірність характеризує наскільки ймовірним є те, що з  $X$  випливає  $Y$ .

$c(X \rightarrow Y) = \text{кількість транзакцій, що містять } X \text{ і } Y /$   
 $\text{кількість транзакцій, що містять } X$

Наприклад, 17,5 % транзакцій, що містять вино, також містять сир.

$$\text{Confidence} = \frac{7}{40} = 17,5\%$$



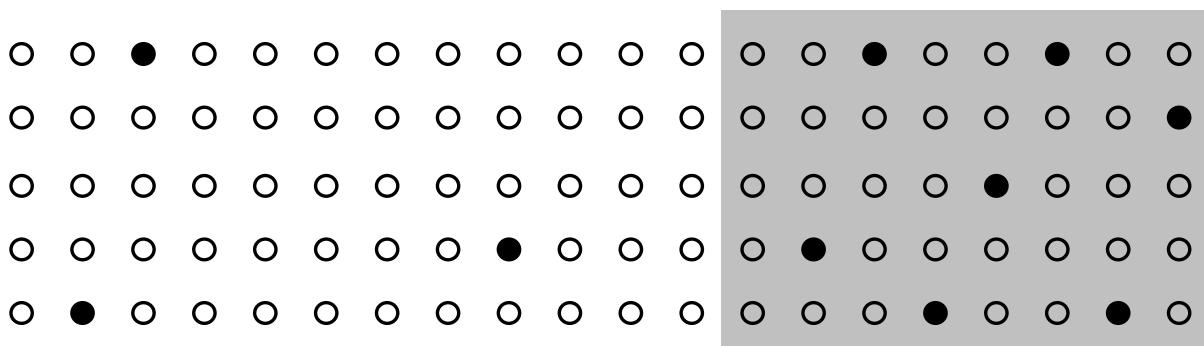
*Ліфт* ( $l$ , lift) – це відношення частоти появи умови в транзакціях, що також містять і наслідок, до частоти появи наслідку в цілому.

$$l(X \rightarrow Y) = c(X \rightarrow Y) / s(Y)$$

Значення ліфта більше за одиницю, показують, що умова з'являється частіше в транзакціях, що містять і наслідок, ніж в інших, що дозволяє підвищити точність таргетування.

Тобто в цьому разі пропозиція акції на сир покупцям вина у 1,75 разів ефективніша, ніж аналогічна акція для всіх покупців.

$$Lift = \frac{17,5\%}{10\%} = 1,75$$



Алгоритми пошуку асоціативних правил призначені для знаходження всіх правил  $X \rightarrow Y$ , причому підтримка і достовірність цих правил повинні знаходитися в певних межах. Якщо підтримка має велике значення, то алгоритми знаходитимуть очевидні правила. Занадто низьке значення підтримки веде до генерації статистично необґрунтованих правил. Значення підтримки за умовчанням – 0,1, однак на практиці зазвичай розглядаються товари, що формують основний прибуток, тобто такі, що купуються, наприклад, не рідше, ніж 4 рази на день. Тоді, маючи сукупність транзакцій за тиждень, мінімальна підтримка розраховується як

$$(4 \text{ рази на день}) * (7 \text{ днів на тиждень}) / \\ (\text{кількість транзакцій}).$$

Значення достовірності за умовчанням – 0,8. На практиці воно визначається шляхом експериментування (зменшуючи базове значення до отримання осмисленої сукупності правил).

## Тема 14. Алгоритми APRIORI та ECLAT

Дані про транзакції зазвичай мають такий вигляд.

Номер транзакції	Найменування елемента	Кількість
1001	A	2
1001	D	3
1001	E	1
1002	A	2
1002	F	1
1003	B	2
1003	A	2
1003	C	2
1003	I	3
...	...	...

Необхідною умовою застосування алгоритму є зведення даних до бінарних значень з наступною нормалізацією. В результаті отримуємо таблицю такої структури.

TID	A	B	C	D	E	F	G	H	I	K	...
1001	1	0	0	1	1	0	0	0	0	0	...
1002	1	0	0	0	0	1	0	0	0	0	...
1003	1	1	1	0	0	0	0	0	1	0	...

Кількість стовпців в таблиці дорівнює кількості елементів, наявних у множені транзакцій D. Кожен запис відповідає транзакції, де у стовпці стоїть 1, якщо елемент наявний у транзакції, і 0 – в іншому випадку. Як видно з таблиці, всі елементи впорядковані в алфавітному порядку (якщо це числа, вони повинні бути впорядковані в числовому порядку).

Алгоритм Apriori працює в два етапи: на першому етапі необхідно знайти сукупності елементів, що часто зустрічаються, на другому – сформувати з них правила. Кількість елементів у сукупності будемо називати розміром сукупності, а сукупність, що складається з  $k$  елементів –  $k$ -елементною сукупністю.

Для зниження розмірності простору пошуку та виявлення найпоширеніших сукупностей Apriori використовує властивість анти-монотонності підтримки: зі зростанням розміру сукупності елементів підтримка зменшується або залишається такою ж. Наприклад, будь-яка транзакція, що містить {Хліб, Масло, Молоко}, також повинна містити {Хліб, Масло}, {Хліб, Молоко}, {Масло, Молоко}, причому зворотне не правильно.

На першому кроці алгоритму підраховуються 1-елементні сукупності, що часто зустрічаються. Наступні кроки складаються з двох частин:

- 1) генерації кандидатів (сукупностей елементів, що часто зустрічаються);
- 2) підрахунку підтримки для кандидатів.

Для генерації кандидатів ( $k$ -елементних наборів) використовуються  $(k-1)$ -елементні сукупності, вже сформовані на попередньому кроці.

Для розрахунку підтримки кандидатів використовується хеш-дерево<sup>4</sup>, внутрішні вузли якого містять хеш-таблиці показчиків на нащадків, а листя – на кандидатів.

Спочатку дерево складається тільки з кореня, що є листом, і не містить ніяких кандидатів-сукупностей. Кожен раз, коли формується новий кандидат, він долучається до кореня дерева, і так доти допоки кількість кандидатів в корені-листі не перевищить певного порогу. Як тільки кількість кандидатів стає більше порогу, корінь перетворюється на хеш-таблицю, тобто стає внутрішнім вузлом, і для нього створюються нащадки-листя. Кожен новий кандидат хешується на внутрішніх вузлах, допоки він не досягне першого вузла-листа, де він і буде зберігатися, допоки кількість сукупностей знову ж таки не перевищить порогу.

---

<sup>4</sup> **Хешування** (гешування, англ. *hashing*) – перетворення вхідного масиву даних довільної довжини на вихідний бітовий рядок фіксованої довжини.

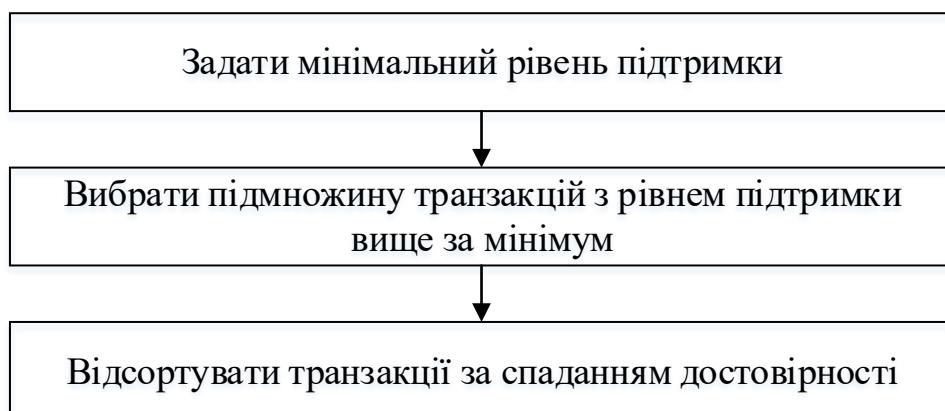
Для підрахунку підтримки кандидатів потрібно «пропустити» кожну транзакцію через дерево та збільшити лічильники для тих кандидатів, чиї елементи також містяться і в транзакції. Потім перевіряється чи задовольняють значення підтримки кандидатів мінімальному порогу.

Для підрахунку достовірності правила досить знати підтримку самої сукупності і множини, що є в умові правила, що також часто зустрічається через властивість анти-монотонності.

Схематично алгоритм можна подати у такий спосіб.



Алгоритм ECLAT є більш простою модифікацією алгоритму Apriori, під час його побудови не розраховується значення ліфта. Схема роботи алгоритму наведена нижче.



## Довідка: основні функції

*itemFrequency* підраховує частоту зустрічальності / підтримку (support) для всіх окрім взятих ознак об'єкта типу itemMatrix

**itemFrequency (x, ...)**

```
## method for signature 'itemMatrix' itemFrequency (x, type)
```

Аргументи

x об'єкт

... пропуск наступних аргументів

type рядок "relative" (за замовчуванням) або "absolute" – відповідно відносна або абсолютна частота

*itemFrequencyPlot* буде графік розподілу частот зустрічальності ознак для об'єктів типу itemMatrix

**itemFrequencyPlot (x, ...)**

```
## method for signature 'itemMatrix' itemFrequencyPlot (x, type = c ("relative",
"absolute"), support = NULL, topN = NULL, population = NULL, popCol = "black",
popLwd = 1, lift = FALSE, horiz = FALSE, names = TRUE, cex.names = par (
"cex.axis"), xlab = NULL, ylab = NULL, mai = NULL, ...)
```

Аргументи

x об'єкт

... пропуск наступних аргументів

type рядок "relative" (за замовчуванням) або "absolute" – відповідно відносна або абсолютна частота

support числове значення; відображаються тільки ознаки з підтримкою не менше support; якщо не задано population – підтримка вираховується з огляду на x, інакше – з population; є абсолютною / відносною залежно від type

topN ціле значення; відображається тільки topN ознак з більшою частотою або підйомом (lift), якщо lift = TRUE; будуться ознаки спаданням значущості

population об'єкт класу x; якщо x – підматриця population, population характеризує частоту народження кожної ознаки, відображену лінією на графіку

popCol колір лінії

popLwd товщина лінії

lift логічний тип; якщо lift = TRUE – графік будується за значенням підйому, інакше – за частотою; функція підйому показує наскільки частіше зустрічається об'єкт в x, ніж в population

horiz логічний тип; якщо horiz = FALSE (за замовчуванням), бар малоється вертикально, інакше – горизонтально

names логічний індикатор, що відповідає за відображення міток

cex.names числове значення; коефіцієнт розширення для назв осей

xlab	рядок; мітка осі x (порожній рядок якщо вісь без мітки)
ylab	рядок; мітка осі y (порожній рядок якщо вісь без мітки)
mai	числовий вектор; розміри графіка в дюймах показує числовий вектор з проміжними точками графіка

*read.transactions* зчитує транзакції з файлу

`read.transactions (file, format = c ("basket", "single"), sep = NULL, cols = NULL, rm.duplicates = FALSE, encoding = "unknown")`

Аргументи

file	ім'я файлу
format	рядок; формат даних
sep	рядок; визначає, як розділені стовпці у файлі, або NULL (за замовчуванням); в форматі basket – може бути регулярним виразом або символом в разі формату single, за замовчуванням роздільниками є пробіли
cols	для формату single параметр подано числовим або символічним вектором довжини 2, що складається з номерів або імен стовпчика відповідно з транзакціями та ідентифікаторами ознак, якщо вектор символічний, то перший рядок файла розпізнається як імена стовпців; для формату basket параметр являє собою число, надаючи номер стовпчика з ID транзакцій. Для параметра NULL дані не містять ID транзакцій.
rm.duplicates	логічний тип; визначає, чи варто видаляти повторювані ознаки в транзакціях
encoding	рядок; визначає кодування, що передається в readlines

*Apriori* аналізує часті сукупності, асоціативні правила

`apriori (data, parameter = NULL, appearance = NULL, control = NULL)`

Аргументи

data	об'єкт класу transactions або будь-яка інша структура, яку можна перетворити в даний клас (наприклад, бінарна матриця)
parameter	об'єкт класу APparameter або іменований список; за замовчуванням – support = 0,1, confidence = 0,8, maxlen = 10
appearance	об'єкт класу APappearance або іменований список; обмежує появу ознак із цим аргументом, за замовчуванням – поява всіх ознак необмежена
control	об'єкт класу APcontrol або іменований список; контролює виконання алгоритму

*Eclat* аналізує часті набори алгоритмом Eclat

`eclat (data, parameter = NULL, control = NULL)`

**Аргументи**

data	об'єкт класу transactions або будь-яка інша структура, що можна перетворити на цей клас
parameter	об'єкт класу ECparameter або іменований список; за замовчуванням – support = 0,1, maxlen = 5
control	об'єкт класу ECcontrol або іменований список; контролює виконання алгоритму

*Support* обчислює значення підтримки для обраних транзакцій бази даних  
**support** (x, transactions, ...)

**## method for signature 'itemMatrix' or 'associations'**

**support** (x, transactions, type = c ("relative", "absolute"), control = NULL)

**Аргументи**

data	множина сукупностей, підтримку яких треба обчислити
...	пропуск наступних аргументів
type	рядок "relative" (за замовчуванням) або "absolute" відповідає обчисленню відносної або абсолютної підтримки
control	іменований список з елементами
method	із зазначенням методу ("tidlists" або "ptree"), логічні аргументи reduce і verbose, що показують чи потрібен докладний висновок

*ruleInduction* генерує всі правила із заданих сукупностей

**ruleInduction** (x, ...)

**## method for signature 'itemsets'**

**ruleInduction** (x, transactions, confidence = 0,8, control = NULL)

**Аргументи**

x	множина сукупностей, правила яких необхідно обчислити
...	пропуск наступних аргументів
transactions	множина транзакцій даних для аналізу наборів
confidence	число; мінімальне значення значущості правил
control	іменований список; вказує метод побудови правил: "apriori" або "ptree", і містить логічні параметри reduce і verbose, що показують віддалені чи невикористовувані ознаки і чи потрібен докладний висновок, що повертає об'єкт класу rules

*sample* генерує вибірку фіксованого розміру з елементів x із перестановками і без

**sample** (x, size, replace = FALSE, prob = NULL, ...)

**Аргументи**

x	множина associations або transactions
...	пропуск наступних аргументів
size	розмір вибірки
replace	логічний тип; відповідає за наявність перестановок
prob	числовий вектор з ймовірними вагами

*random.transactions* генерує випадковий об'єкт transactions, використовуючи різні методи  
*random.transactions* (nItems, nTrans, method = "independent", ..., verbose = FALSE)

Аргументи

nItems	ціле число; кількість ознак
nTrans	ціле число; кількість транзакцій
size	розмір вибірки
method	назва методу: "independent" або "agrawal"
...	пропуск наступних аргументів
verbose	логічний тип; чи потрібен висновок звіту щодо виконання функції

*interestMeasure* генерує вибірку фіксованого розміру з елементів x із перестановками і без

*interestMeasure* (x, method, transactions = NULL, reuse = TRUE, ...)

Аргументи

x	множина сукупностей або правил
method	рядок або вектор рядків
transactions	множина транзакцій для аналізу асоціацій
reuse	логічний тип; необхідна інформація в слоті для перерахунку мір
...	пропуск наступних аргументів

*Dissimilarity* обчислює відстані між векторами бінарної матриці: transactions або associations

*dissimilarity* (x, y = NULL, method = NULL, args = NULL, ...)

## method for signature 'itemMatrix' or 'associations'

*dissimilarity* (x, y = NULL, method = NULL, args = NULL, which = "Transactions")

## method for signature 'matrix'

*dissimilarity* (x, y = NULL, method = NULL, args = NULL)

Аргументи

x	множина елементів
y	NULL або множина для обчислення відстаней
method	рядок – метод, що визначає поняття відстані; "affinity", "cosine", "dice", "euclidean", "jacard" (за замовчуванням), "matching", "pearson"; додатково для асоціацій – "toivonen", "gupta"
args	спісок аргументів відповідного методу
...	пропуск наступних аргументів
which	рядок, що визначає, чи будуть відстані між транзакціями обчислені (за замовчуванням) або бінарними ознаками ("items")

## Лабораторна робота 12

### Association Rules

#### Download the data

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##     filter, lag

## The following objects are masked from 'package:base':
##     intersect, setdiff, setequal, union

set.seed(123)
#setwd('D:/ML')
#install.packages('arules')
library(arules)

## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':
##     recode

## The following objects are masked from 'package:base':
##     abbreviate, write

f <- read.transactions('mini-market.csv', sep = ',', rm.duplicates = TRUE)

## distribution of transactions with duplicates:
## items
##    2     3     4     5     6     7
##    2    11    31   112   256  1197

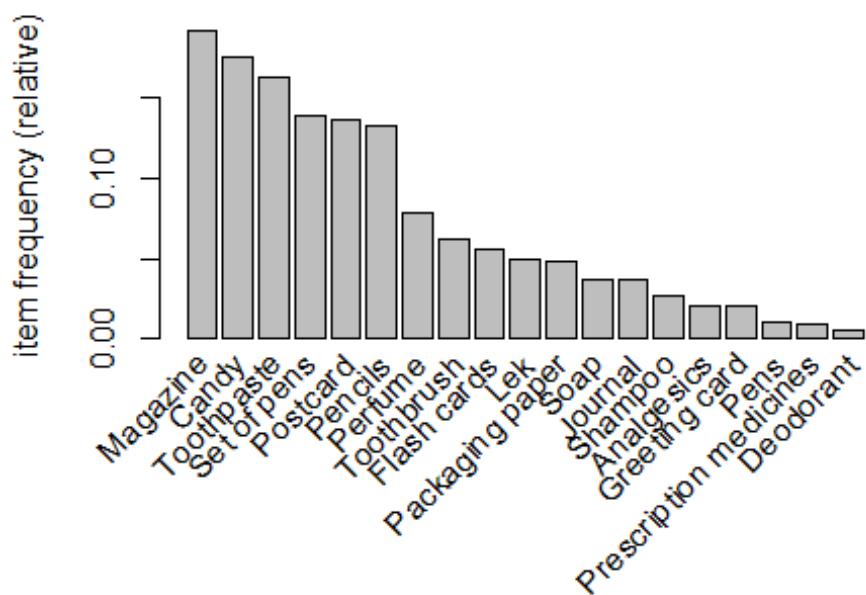
summary(f)

## transactions as itemMatrix in sparse format with
## 1611 rows (elements/itemsets/transactions) and
## 1640 columns (items) and a density of 0.001462506
## 

## most frequent items:
##     Magazine      Candy  Toothpaste Set of pens     Postcard      (Other)
##            307        282       261        223        218        2573
## 
```

```
## element (itemset/transaction) length distribution:
## sizes
##   2    3    4    5    6    7    9   10
## 1197  256  112   31   11    2    1    1
##
##      Min. 1st Qu. Median     Mean 3rd Qu.     Max.
##    2.000  2.000  2.000   2.399  3.000  10.000
##
## includes extended item information - examples:
##   labels
## 1    1
## 2   10
## 3  100

itemFrequencyPlot(f, topN = 19)
```



**Висновок:** для роботи з асоціативними правилами дані зчитуються в форматі транзакцій – кількість стовпців в таблиці дорівнює кількості елементів, наявних у множені транзакцій D. Найчастішими покупками є журнали, цукерки та зубна паста.

## Eclat

```
model_eclat = eclat(data = f, parameter = list(support = 0.017, minlen = 2))
##
## Eclat
## 
## parameter specification:
##   tidLists support minlen maxlen           target   ext
##   FALSE     0.017       2      10 frequent itemsets FALSE
```

```

## 
## algorithmic control:
##   sparse sort verbose
##     7    -2    TRUE
##
## Absolute minimum support count: 27
##
## create itemset ...
## set transactions ...[1640 item(s), 1611 transaction(s)] done [0.00s].
## sorting and recoding items ... [16 item(s)] done [0.00s].
## creating sparse bit matrix ... [16 row(s), 1611 column(s)] done [0.00s]
.
## writing ... [10 set(s)] done [0.00s].
## Creating S4 object ... done [0.00s].

# Visualising the results
inspect(sort(model_eclat, by = 'support')[1:10])

##      items          support    count
## [1] {Candy,Toothpaste} 0.03600248 58
## [2] {Magazine,Postcard} 0.03476102 56
## [3] {Candy,Postcard} 0.03289882 53
## [4] {Candy,Magazine} 0.03289882 53
## [5] {Candy,Pencils} 0.03103662 50
## [6] {Pencils,Postcard} 0.02855369 46
## [7] {Magazine,Toothpaste} 0.02793296 45
## [8] {Postcard,Toothpaste} 0.02482930 40
## [9] {Magazine,Pencils} 0.02296710 37
## [10] {Pencils,Toothpaste} 0.01986344 32

```

**Висновок:** значення підтримки за умовчанням – 0,1, однак на практиці зазвичай розглядаються товари, що формують основний прибуток і купуються, наприклад, не рідше, ніж 4 рази на день. Тоді, маючи сукупність транзакцій за тиждень, мінімальна підтримка розраховується як (4 рази в день) \* (7 днів на тиждень) / (кількість транзакцій = 1610) = 0,017. Мінімальна кількість товарів у сукупності – 2. Найчастішими сукупностями є цукерки та зубна паста; журнали та листівки, цукерки та листівки.

## Apriori

```

model_ap = apriori(data = f, parameter = list(support = 0.01, confidence =
0.2))

## Apriori
##
## Parameter specification:
##   confidence minval smax arem  aval originalSupport maxtime support minl
## en
##     0.2    0.1    1 none FALSE           TRUE      5    0.01
## 1
##   maxlen target  ext
##     10  rules FALSE
## 
```

```

## Algorithmic control:
##   filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 16
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1640 item(s), 1611 transaction(s)] done [0.00s].
## sorting and recoding items ... [17 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object ... done [0.00s].

# Visualising the results
inspect(sort(model_ap, by = 'lift')[1:10])

##      lhs                  rhs      support  confidence lift
## [1] {Magazine,Pencils} => {Postcard} 0.01241465 0.5405405 3.994545
## [2] {Candy,Magazine}    => {Postcard} 0.01489758 0.4528302 3.346374
## [3] {Toothbrush}        => {Perfume}  0.01489758 0.2424242 3.099567
## [4] {Magazine,Postcard} => {Pencils}   0.01241465 0.3571429 2.713949
## [5] {Magazine,Postcard} => {Candy}    0.01489758 0.4285714 2.448328
## [6] {Candy,Postcard}    => {Magazine} 0.01489758 0.4528302 2.376252
## [7] {Pencils,Postcard}  => {Magazine} 0.01241465 0.4347826 2.281547
## [8] {Pencils}           => {Postcard} 0.02855369 0.2169811 1.603471
## [9] {Postcard}          => {Pencils}  0.02855369 0.2110092 1.603471
## [10] {Postcard}         => {Candy}   0.03289882 0.2431193 1.388883
##      count
## [1] 20
## [2] 24
## [3] 24
## [4] 20
## [5] 24
## [6] 24
## [7] 20
## [8] 46
## [9] 46
## [10] 53

```

**Висновок:** значення достовірності за умовчанням – 0,8. На практиці воно визначається шляхом експериментування (зменшуючи базове значення доки не буде отримано осмисленої сукупності правил). Найчастішими наборами за заданої сукупності параметрів є журнали та олівці, цукерки та журнали, зубна щітка.

## Graph

```

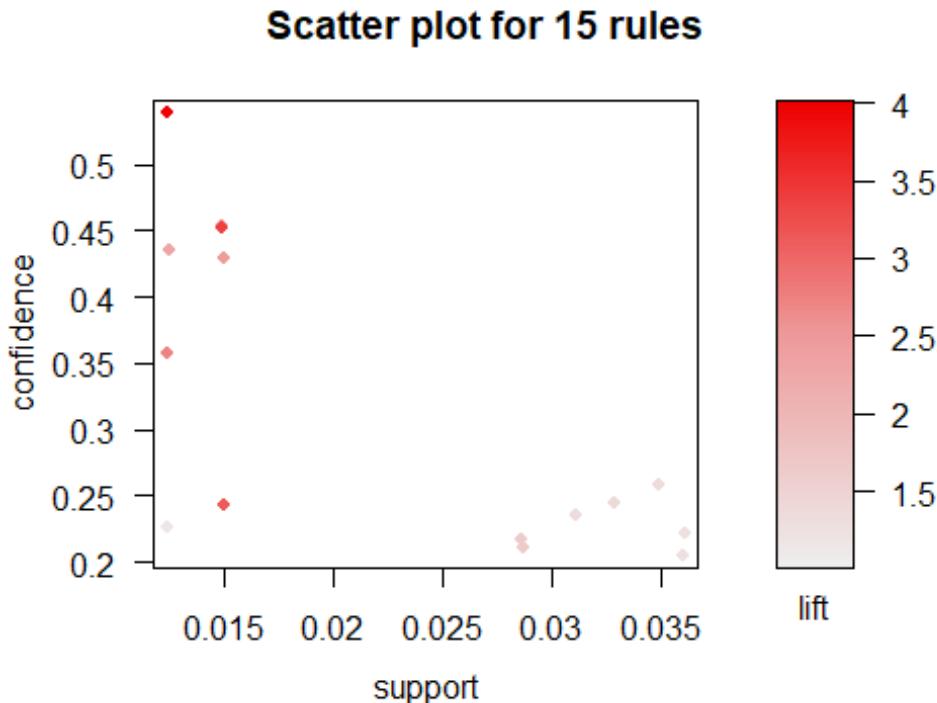
library(arulesViz)

## Loading required package: grid

## Registered S3 method overwritten by 'seriation':
##   method      from
##   reorder.hclust gclus

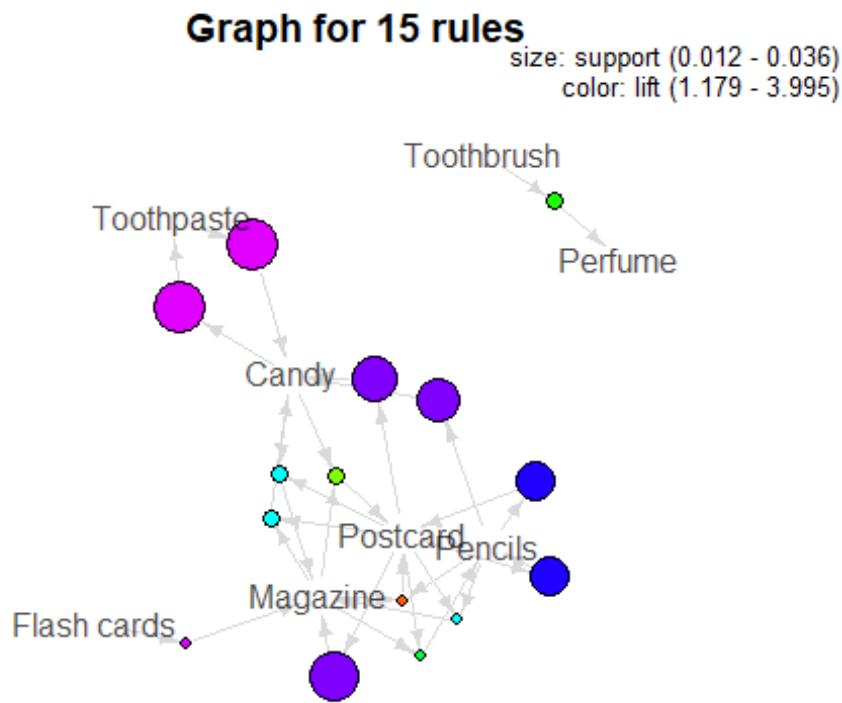
```

```
plot(model_ap, measure = c("support", "confidence"), shading = "lift")
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



Висновок: функція `plot()` з пакету `arulesViz` дозволяє отримувати різні форми візуалізації синтезованих правил, зокрема аналіз мінливості їх якості.

```
plot(head(sort(model_ap, by = "support"), 30), method = "graph",
      control = list(nodeCol = rainbow(16),
                     edgeCol = grey(.85), alpha = 1))
```



**Висновок:** результати побудови правил зручно подати на графі. Граф дозволяє побачити ланцюжка товарів, які частіше купують один з одним, простежити закономірності формування наборів товарів. Метод “graph” функції plot () показує правила та їх складові ознаки у вигляді графа, розмір вузлів якого пропорційний рівню підтримки кожного правила.

### Довідка: Apriori

```
apriori(data, parameter = NULL, appearance = NULL, control = NULL)
```

#### Arguments

- Data** object of class **transactions** or any data structure which can be coerced into **transactions** (e. g., a binary matrix or data.frame).
- parameter** object of class **APparameter** or named list. The default behavior is to mine rules with minimum support of 0.1, minimum confidence of 0.18, maximum of 10 items (maxlen), and a maximal time for subset checking of 5 seconds (maxtime).
- appearance** object of class **APappearance** or named list. With this argument item appearance can be restricted (implements rule templates). By default all items can appear unrestricted.
- control** object of class **APcontrol** or named list. Controls the algorithmic performance of the mining algorithm (item sorting, report progress (verbose), etc.)

### **Питання для самоперевірки**

1. Метод пошуку асоціацій.
2. Особливості алгоритму Eclat.
3. Особливості алгоритму Apriori.
4. Поняття достовірності.
5. Поняття підтримки.
6. Поняття ліфта.

### **Самостійна робота 5**

На основі даних про продажі побудувати моделі пошуку асоціативних правил, проаналізувати результати.

## РОЗДІЛ 6. ОБРОБКА ПРИРОДНОЇ МОВИ

### Тема 15. Модель «мішок слів»

Модель тексту «мішок слів» (bag-of-words) була запропонована 1975 року Дж. Солтоном. Модель використовується для попередньої обробки текстів з метою подальшого застосування алгоритмів класифікації до сформованого корпусу. На сьогодні вона є однією з найбільш поширених в алгоритмах машинного навчання для обробки природної мови.

Найпростіша модель тексту «мішок слів» є сумативною множиною слів, складових тексту; вона дозволяє виявити слова, складові тексту і визначити частоту, з якою вони в тексті зустрічаються. У моделі враховується тільки кількість входжень конкретних слів в початковому тексті, водночас ігноруються:

- порядок слів у документі;
- морфологічні форми подання слів.

В результаті застосування моделі формується корпус текстів, в якому кожен документ (запис) виглядає як невпорядкована сукупність слів без відомостей про зв'язки між ними. Його можна подати у вигляді матриці, кожен рядок в якій відповідає окремому документу, а кожен стовпець – певному слову. Комірка на перетині рядка й стовпця містить кількість входжень слова у відповідний документ.

В R для роботи з моделлю «мішка слів» використовується бібліотека *tm*. Як об'єкт вона працює з лінгвістичним корпусом першого порядку – колекцією документів, об'єднаних загальною ознакою. Для того, щоб скласти корпус спочатку потрібно перетворити тексти на вектор, кожен елемент якого являє собою окремий документ.

Перш ніж ми зможемо працювати зі сформованим корпусом, необхідно його очистити. Цей процес містить такі етапи:

- 1) переводимо всі букви в нижній регістр, щоб уникнути повторів слів в різному регистрі;
- 2) видаляємо числа з текстів;
- 3) видаляємо пунктуацію;
- 4) видаляємо «стоп-слова» – поширені слова, такі як «this», «that», «and», «so», «on» («цей», «той», «і», «тому», «на»), що не містять суттєвої з точки зору подальшої класифікації інформації;
- 5) проводимо стемінгування – зведення слів до єдиної словоформи зі збереженням лише кореня слова;
- 6) видаляємо зайві пробіли, що утворилися із видаленням чисел, стоп-слів та інших перетворень.

В результаті отримуємо розріджену матрицю, в якій кількість рядків відповідає кількості відгуків, а кількість стовпців – унікальним словами, що зустрічалися у всіх текстах. В окремому рядку матриці ненульове значення відповідає кількості входжень цього слова в документ.

Залежно від типу класифікатора, що планується використовувати в кожному конкретному випадку, ухвалюється рішення про скорочення розмірності отриманої матриці. Зокрема, для випадкового лісу, що являє собою класифікатор з високою нелінійністю та дисперсією, потрібно, щоб кількість елементів в навчальній вибірці значно перевищувало розмірність. Лінійні моделі менш вимогливі щодо цього, вони можуть працювати, навіть якщо обсяг навчальної вибірки значно менше розмірності. Для скорочення розмірності матриці можна видалити найменш частотні слова. Однак видалення всього 0,1 % найменш частотних слів може привести до істотного скорочення розмірності, тому налаштування цього параметру вимагає особливої уваги.

Для побудови класифікатора до сформованої матриці необхідно додати стовпець ендогенної змінної, у такий спосіб на базі корпусу текстів формується матриця «документ-термін» (де термін – класифікаційна ознака). Ця матриця і називається «мішком слів».

Крім перерахованих, до корпусу текстів можуть бути застосовані додаткові перетворення. Наприклад, TF-IDF (term frequency – inverse document frequency, частота слова – зворотна частота документа) – метод, що збільшує ваги слів, що часто зустрічаються в цьому документі, і зменшує ваги слів, що часто зустрічаються в багатьох документах.

Крім того, часто доцільно застосувати N-грами (N-gram), в цьому разі видаляти «стоп-слова» не можна. N-грама – це послідовність з N слів. Наприклад, біграми складаються з двох слів: «cat ate», «ate my», «my precious», «precious homework» («кіт з'їв», «з'їв мою», «мою дорогоцінну», «дорогоцінну роботу»); триграми складаються з трьох слів: «cat ate my», «ate my homework», «my precious homework» («кіт з'їв мою», «з'їв мою роботу», «мою дорогоцінну роботу»); чотириграми складаються з чотирьох слів тощо.

Ефективність застосування N-грам розглянемо на такому прикладі: візьмемо фразу «movie not good» («фільм не хороший»). Очевидно, що вона має негативну тональність. Проте якщо розглядати кожне слово окремо, це визначити неможливо. Крім того, модель, ймовірно, «вивчить», що слово «good» («хороший») має позитивну тональність, що в цьому разі неправильно. За допомогою біграм моделі, ймовірно, «вивчить», що вираз «not good» («не хороший») має негативну тональність.

Розглянемо більш складний приклад зі сторінки, присвяченій аналізу тональності тексту, на сайті Стенфордського університету: «This movie was actually neither that funny, nor super witty» («Насправді, цей фільм не був ні дуже смішним, ні супер дотепним»). В цьому випадку біграми «that funny» («дуже смішний») і «super witty» («супер дотепний») дадуть неправильний результат. Тут необхідні принаймні триграми, щоб правильно тлумачити тональність висловлювань «neither that funny» («ні дуже смішний») і «nor super witty» («ні супер дотепний»). Втім подібні фрази зустрічаються не дуже часто. Якщо ми використовуємо обмежену кількість ознак або регуляризацію, вони неістотно впливають на модель.

## Лабораторна робота 13

# Natural Language Processing

### Importing the dataset

Вихідний файл повинен бути в форматі .tsv (розділовий знак – табуляція), а не .csv (розділовий знак – кома), тому що вихідний текст може містити коми, як частини тексту, що можуть бути інтерпретовані алгоритмом як розподільні стовпців. Для читання файлів такого типу використовується функція `read.delim`.

#Параметр `quote = ''` дозволяє позбутися лапок в тексті вже на етапі читання файлу

#Параметр `stringsAsFactors = FALSE` дозволяє не розпізнавати зміст відгуків як значення факторних змінних

```
dataset = read.delim('Restaurant_Reviews.tsv', quote = '', stringsAsFactors = FALSE)
```

### Cleaning the texts

Очищення тексту має на меті: 1) скорочення розмірності даних; 2) відбір найбільш релевантних змінних.

```
#install.packages('tm')
#install.packages('SnowballC')
library(tm)

## Loading required package: NLP
```

```
library(SnowballC)
```

#Для початку роботи з відгуками слід створити корпус текстів

```
corpus = VCorpus(VectorSource(dataset$Review))
```

#Для роботи з корпусом використовуємо функцію `tm_map`  
#Переводимо всі букви в нижній регістр, щоб уникнути повторів слів в різному регістрі

```
corpus = tm_map(corpus, content_transformer(tolower))
```

#Видаляємо числа з текстів

```
corpus = tm_map(corpus, removeNumbers)
```

#Видаляємо пунктуацію

```
corpus = tm_map(corpus, removePunctuation)
```

#Видаляємо "стоп-слова", таких як `this`, `why`, `etc.`, використовуємо для цього функцію `"stopwords"`; ця функція з бібліотеки `SnowballC`

```
corpus = tm_map(corpus, removeWords, stopwords())
```

```
#Проводимо стеммірування, зберігаючи тільки корінь слова
corpus = tm_map(corpus, stemDocument)

#Видаляємо зайві пробіли, що утворилися під час видалення стоп-слів та
#інших перетворень
corpus = tm_map(corpus, stripWhitespace)
```

**Висновок:** в результаті отримуємо розріджену матрицю, в якій кількість рядків відповідає кількості відгуків, а кількість стовпців – унікальним словами, що зустрічалися у всіх текстах. В окремому рядку матриці 1 відповідає наявності цього слова в цьому відгуку.

## Creating the Bag of Words model

```
#Для створення моделі "мішок слів" в якості незалежної змінної виступає
створений вище корпус текстів. Конвертуємо його в матрицю
dtm = DocumentTermMatrix(corpus)
```

#Зверненням до матриці – dtm – можна подивитися її характеристики. У цьому разі ми отримали матрицю розмірністю 1000\*1577, із 100 % розрідженністю. Для скорочення її розмірності і розрідженності видалимо найменш частотні слова. Залишимо 99,9 % найбільш частотних слів. В результаті, розмірність матриці скоротилася до 1000\*691, розрідженість – 99 %
dtm = removeSparseTerms(dtm, 0.999)

```
#Для побудови класифікатора отриману матрицю необхідно перетворити в дата-
фрейм – as.data.frame. Водночас потрібно уточнити, що ми звертаємося до
неї, як до матриці – as.matrix
ds = as.data.frame(as.matrix(dtm))
```

```
#Додамо стовпець для ендогенної змінної, в якій зафікований характер
відгуку – позитивний чи негативний
ds$Liked = dataset$Liked
```

```
#Оголосимо тип залежної змінної – якісна (factor)
ds$Liked = factor(ds$Liked, levels = c(0, 1))
```

## Splitting the dataset into the Training set and Test set

```
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(ds$Liked, SplitRatio = 0.8)
training_set = subset(ds, split == TRUE)
test_set = subset(ds, split == FALSE)
```

**Висновок:** підготований датасет розподілено на навчальну та тестову вибірки.

## Fitting Random Forest Classification to the Training set

```
# install.packages('randomForest')
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
classifier = randomForest(x = training_set[-692],
                           y = training_set$Liked,
                           ntree = 10)
```

Висновок: проведено навчання моделі випадкового лісу.

## Predicting the Test set results

```
y_pred = predict(classifier, newdata = test_set[-692])
```

Висновок: визначено класи об'єктів.

## Making the Confusion Matrix

```
cm = table(test_set[, 692], y_pred)
cm

##     y_pred
##     0   1
##   0 82 18
##   1 23 77
```

Висновок: точність моделі –  $(82+77) / 200 = 79,5\%$ , частка невірно класифікованих випадків –  $(18+23) / 200 = 20,5\%$ . Чутливість –  $77 / (23+77) = 77\%$ , специфічність –  $82 / (82+18) = 82\%$ , тобто модель більш чутлива до виявлення негативних відгуків.

## Довідка: tm\_map

```
## S3 method for class 'PCorpus'
tm_map(x, FUN, ...)
## S3 method for class 'SimpleCorpus'
tm_map(x, FUN, ...)
## S3 method for class 'VCorpus'
tm_map(x, FUN, ..., lazy = FALSE)
```

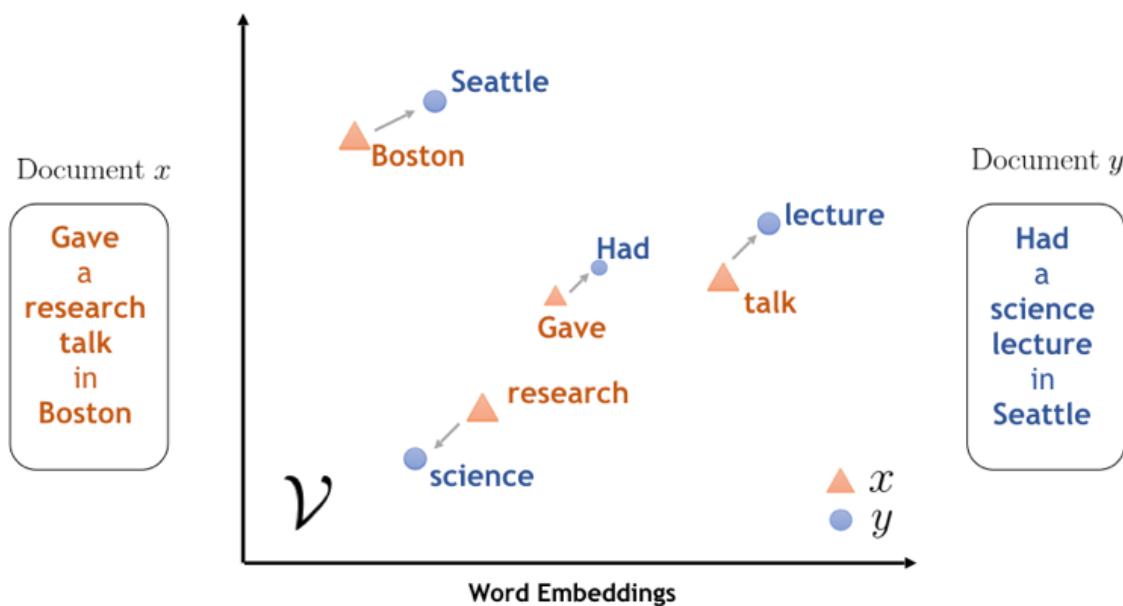
### Arguments

- x A corpus.
- FUN a transformation function taking a text document (a character vector when x is a SimpleCorpus) as input and returning a text document (a character vector of the same length as the input vector for SimpleCorpus). The function `content_transformer` can be used to create a wrapper to get and set the content of text documents.
- ... arguments to FUN.
- lazy a logical. Lazy mappings are mappings which are delayed until the content is accessed. It is useful for large corpora if only few documents will be accessed. In such a case it avoids the computationally expensive application of the mapping to all elements in the corpus.

## Тема 16. Моделі з урахуванням семантики

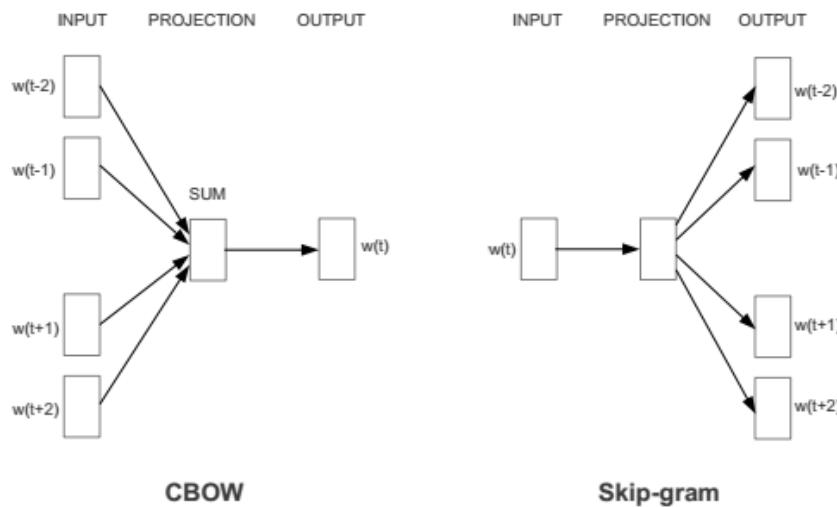
Основним недоліком унітарного кодування (коли кожне слово ставиться у відповідність 0 або 1 в залежності від його наявності в документі), що використовується в моделі «мішок слів», є брак можливості обліку схожості, близькості різних слів. Найпростіший спосіб описати схожість векторів – це обчислити їх скалярний добуток, проте у разі унітарного кодування скалярний добуток дорівнює нулю.

Для подолання цієї проблеми використовується технологія Word2vec – сукупність моделей для аналізу семантики природних мов, заснованих на векторному поданні слів. В якості вхідних даних Word2vec використовує великий текстовий корпус і співставляє кожному слову вектор, видаючи координати слів на вихіді. Векторне подання ґрунтуються на контекстній близькості: слова, що зустрічаються в тексті поруч з однаковими словами (що мають схоже значення), у векторному поданні матимуть близькі координати векторів-слів. У такий спосіб отримані на вихіді векторні уявлення слів дозволяють обчислювати «семантичну відстань» між словами.



Технологія Word2vec працює з нейронними мережами.

Для Word2vec розроблені два основні алгоритми навчання: CBoW (Continuous Bag of Words, «неперервний мішок слів») та Skip-gram. CBoW – архітектура, що передбачає слово з огляду на навколошній контекст. Завданням методу CBOW є передбачення слова на підставі прилеглих слів. У Skip-gram зворотна задача – передбачення сукупності прилеглих слів на підставі поданого слова. Порядок слів контексту не впливає на результат у жодному з цих алгоритмів.



\*  $w(t)$  – подане слово, а  $w(t-2), w(t-1)$  тощо – прилеглі слова.

### Питання для самоперевірки

1. Особливості моделі «мішок слів».
2. Етапи формування корпусу текстів.
3. Методи скорочення розмірності розрідженої матриці частотності слів.
4. Особливості TF-IDF методу.
5. Особливості застосування N-грам.
6. Особливості моделі Word2vec.
7. Алгоритми CBoW і Skip-gram.

### Самостійна робота 6

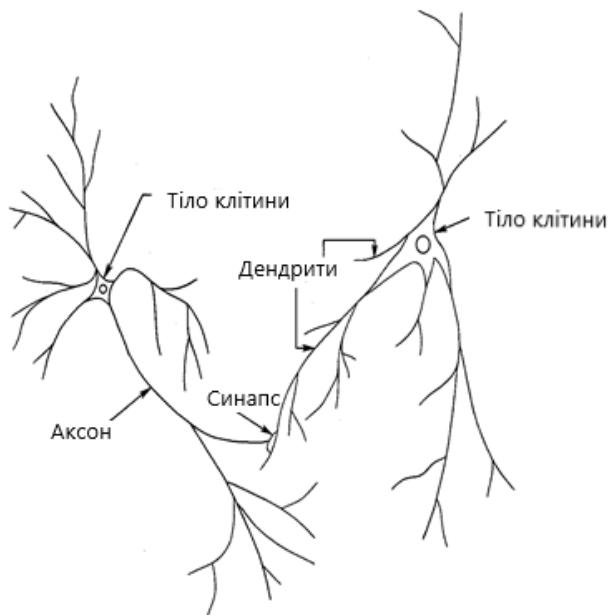
На основі відгуків про товари побудувати модель оцінки його привабливості, зробити висновки про якість моделі.

## РОЗДІЛ 7. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

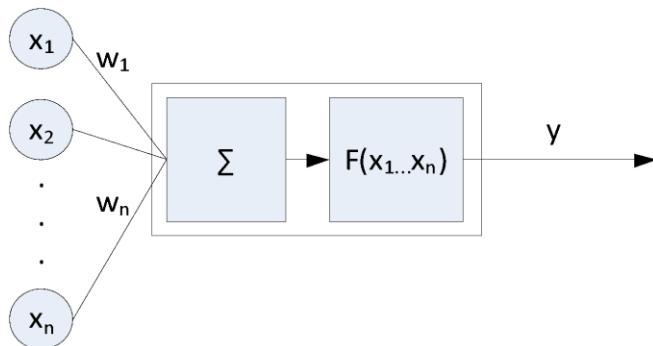
### Тема 17. Теоретичні основи нейронних мереж

Нейронні мережі (НМ) – розподілені та паралельні системи, здатні до адаптивного навчання шляхом реакції на зовнішні впливи. Цей напрямок інспіровано біологією, НМ насамперед імітують роботу мозку.

Мозок містить понад тисячу мільярдів обчислювальних елементів, що називають *нейронами*. Спрощено нейрон складається з трьох елементів: тіла клітини, дендритів і аксона. Дендрити отримують сигнали від нервових клітин через контакти, так звані *синапси*. Звідси сигнали проходять в тіло клітини, де вони сумуються з іншими сигналами. Якщо сумарний сигнал протягом короткого проміжку часу є досить великим, клітина збуджується, виробляючи в аксоні імпульс, що передається до інших клітин.



**Штучний нейрон** імітує властивості біологічного нейрона. До входу штучного нейрона надходить деяка множина сигналів, кожен з яких є виходом іншого нейрона. Кожен вхід помножується на відповідну вагу (аналогічну синаптичній силі) і всі добутки сумуються, визначаючи рівень активації нейрона. Отже, основою побудови мережі є елементарний перетворювач, так званий штучний нейрон.



Нейрон складається зі зваженого суматора і нелінійного елемента.

Функціонування нейрона визначається співвідношеннями

$$s = \sum w_i x_i$$

$$y = f(s, b)$$

де  $x_i$  – вхідні сигнали (сукупність всіх вхідних сигналів нейрона утворює вектор  $x$ );

$w_i$  – вагові коефіцієнти;

$s$  – зважена сума вхідних сигналів (є аргументом функції активації);

$b$  – пороговий рівень нейрона;

$f$  – функція активації.

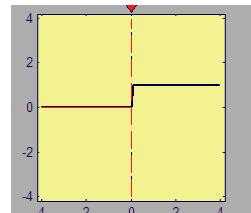
**Види функцій активації.** Розглянемо основні види функцій активації, що поширені в штучних НМ.

*Жорстка сходинка*

$$y = \begin{cases} 0, & s < b \\ 1, & s \geq b \end{cases}$$

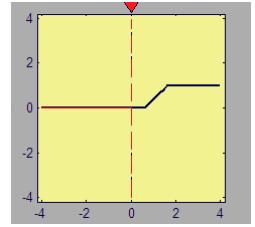
Використовується в класичному формальному нейроні.

Ця функція надмірно спрощена і не дозволяє моделювати схеми з неперервними сигналами. Брак першої похідної ускладнює застосування градієнтних методів для навчання таких нейронів.



*Полога сходинка*

$$y = \begin{cases} 0, & s < b \\ \frac{s-b}{\Delta}, & b \leq s < b + \Delta \\ 1, & s \geq b + \Delta \end{cases}$$

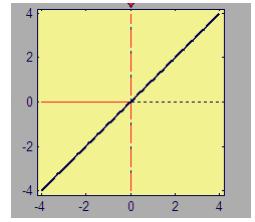


Розраховується легко, але має розривну першу похідну, тому також не використовується під час побудови багатошарових мереж.

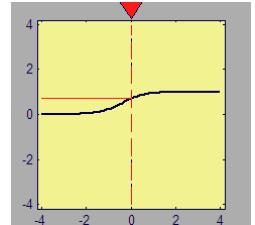
*Лінійна функція*

$$y = ks, \quad k = \text{const}$$

Застосовується для тих моделей мереж, де не потрібно послідовне з'єднання шарів нейронів один за одним.

*Логістична функція*

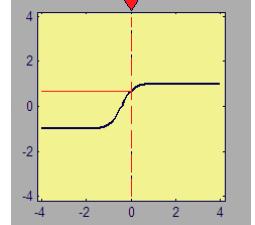
$$y = \frac{1}{1 + e^{-s}}$$



Застосовується для багатошарових мереж з неперервними сигналами. Неперервність першої похідної дозволяє навчати мережу градієнтними методами (наприклад, методом зворотного поширення помилки). Ця функція – стискає, тобто для малих значень  $s$  коефіцієнт передачі  $k = y/s$  великий, для великих значень він знижується.

*Гіперболічний тангенс*

$$y = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

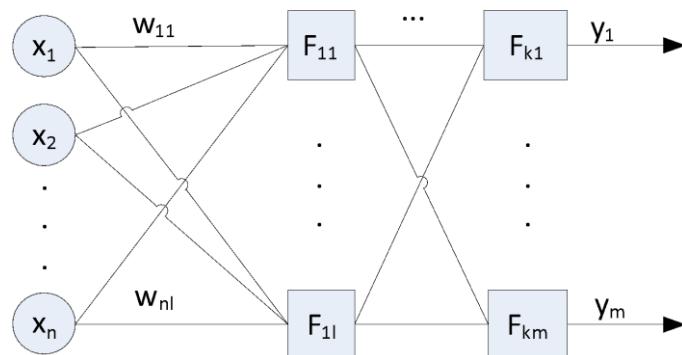


Застосовується для мереж з неперервними сигналами. Функція симетрична щодо точки  $(0,0)$  – перевага порівняно із логістичною кривою. Похідна також неперервна та виражається через саму функцію.

Окрім розглянутих вище функцій активації, останнім часом все більшої популярності набуває блок лінійної ректифікації (rectified linear unit, ReLU). Ця функція приймає значення рівні нулю для негативних значень  $x$  і лінійно зростає для позитивних, тобто вона задається так:  $f(x) = \max(0, x)$ .

**Штучні нейронні мережі (НМ)** є мережею елементів – штучних нейронів – пов’язаних між собою синаптичними вагами. Мережа обробляє вхідну інформацію та генерує сукупність вихідних сигналів. Процес налаштування синаптичних ваг називають навчанням мережі.

Нейромережа складається з декількох шарів: вхідного, внутрішніх (прихованих) та вихідного шарів. Вхідний шар реалізує зв’язок з вхідними даними, вихідний – з вихідними. На кожному шарі міститься декілька нейронів, з’єднаних між собою вагами.



Незважаючи на досить поверхневу подібність, штучні нейронні мережі демонструють властивості, притаманні мозку:

1. *Вчитися.* Обравши одну з моделей, створивши мережу і виконав її навчання, НМ здатна вирішувати відповідні завдання;
2. *Здатність до узагальнення.* Після навчання мережа стає нечутливою до малих змін вхідних сигналів (шуму або варіацій вхідних образів);
3. *Здатність до абстрагування.* Якщо надати мережі кілька спотворених варіантів вхідного образу, то вона може відтворити на виході ідеальний образ.

**Вибір архітектури мережі.** Під час проектування нейронної мережі необхідно вирішити питання про кількість шарів і кількість елементів (нейронів) на кожному шарі.

Під час визначення розмірності прихованого шару наявний компроміс між точністю і узагальнюючою здатністю мережі, що можна оптимізувати за допомогою вибору кількості прихованих елементів. З одного боку воно повинно бути достатнім для вирішення поставленого завдання з необхідною точністю для збереження інформації, що міститься в навчальній вибірці, а з іншого – не повинно бути занадто великим, щоб забезпечити необхідну узагальнюючу здатність. Однак не існує простого способу для визначення необхідної кількості прихованих елементів.

Важливо вирішити, чи буде мережа такою, що звужується (кількість елементів в прихованому шарі менше, ніж кількість вхідних елементів) або навпаки, такою, що розширюється. Так, для вирішення завдання вилучення інформації із входів з метою узагальнення або зниження розмірності масиву даних необхідно використовувати мережу, що звужується. Однак, чим вище вимоги до розрізнення близьких вхідних векторів, тим більше повинна **розширюватися** мережа, і тим нижче її узагальнююча здатність<sup>5</sup>.

Для обчислення верхньої межі кількості прихованих елементів може бути використана теорема Колмогорова, що стверджує, що будь-яка функція  $n$  змінних може бути подана як суперпозиція  $2n + 1$  одновимірних функцій. Інакше кажучи, немає ніякого сенсу вибирати кількість прихованих елементів більше за подвоєну кількість вхідних елементів.

---

<sup>5</sup> Вагові коефіцієнти можна розглядати як додаткові члени або параметри функції. Так, якщо в модель необхідно додати член вищого порядку, слід збільшити кількість вагових коефіцієнтів в мережі. Так само, як криву з однією точкою повороту можна моделювати додаванням до лінійної моделі члена  $x^2$  (для двох точок повороту слід додати член  $x^3$ ), єдиний прихований шар здатний моделювати одну точку повороту в даних. Теж справедливо і для границь в задачах класифікації – чим більше кривих і точок повороту у вхідних даних, тим більша кількість прихованих елементів потрібна для побудови моделі. Але не можна просто вибрати теоретичний максимум кількості вагових коефіцієнтів, бо в цьому разі мережа навчиться мати справу тільки з тими даними, що надавалися в процесі тренування, тому узагальнююча здатність мережі буде слабкою. У цьому сенсі вибір розміру прихованого шару залежить від розв'язуваної задачі.

Крім того, необхідно мати достатню кількість даних для тренування мережі з обраною кількістю вагових коефіцієнтів. Кількість тренувальних прикладів має бути приблизно рівною кількості ваг мережі, помноженій на зворотну величину помилки. Ця залежність описується формулою

$$w < n / e.$$

Під час визначення *кількості прихованих шарів* слід пам'ятати, що мережі, що складаються тільки з вхідного і вихідного шарів, здатні моделювати виключно лінійні функції. Такі архітектури використовуються дуже рідко, що не завжди виправдано, бо вони забезпечують універсальну лінійну апроксимацію. Доти допоки немає впевненості в тому, що задача істотно нелінійна, має сенс намагатися вирішити її за допомогою одношарової мережі.

З використанням теореми Колмогорова показано, що будь-яка функція може бути апроксимована з використанням максимум чотирьох шарів. Однак для вирішення більшості практичних економічних задач досить одного, іноді двох прихованых шарів. Причина такої невідповідності теорії і практики в тому, що складність реальних проблем набагато менша, ніж це теоретично можливо.

Для навчання багатошарових мереж використовують *функції активації*, що мають *неперервну першу похідну*, що повинні мати такі властивості: по-перше, стискати вхідні значення в фіксований діапазон, подруге, їх похідна повинна слабко змінюватися на кожному з країв діапазону і сильно змінюватися в середині діапазону. В якості таких функцій використовуються або логістична функція, або функція гіперболічний тангенс.

Логістична функція несиметрична, що призводить до збільшення часу тренування, гіперболічний тангенс – непарна функція, що трохи зменшує швидкість навчання. Обидві функції називаються сигмоїдальними завдяки їх S-образній формі.

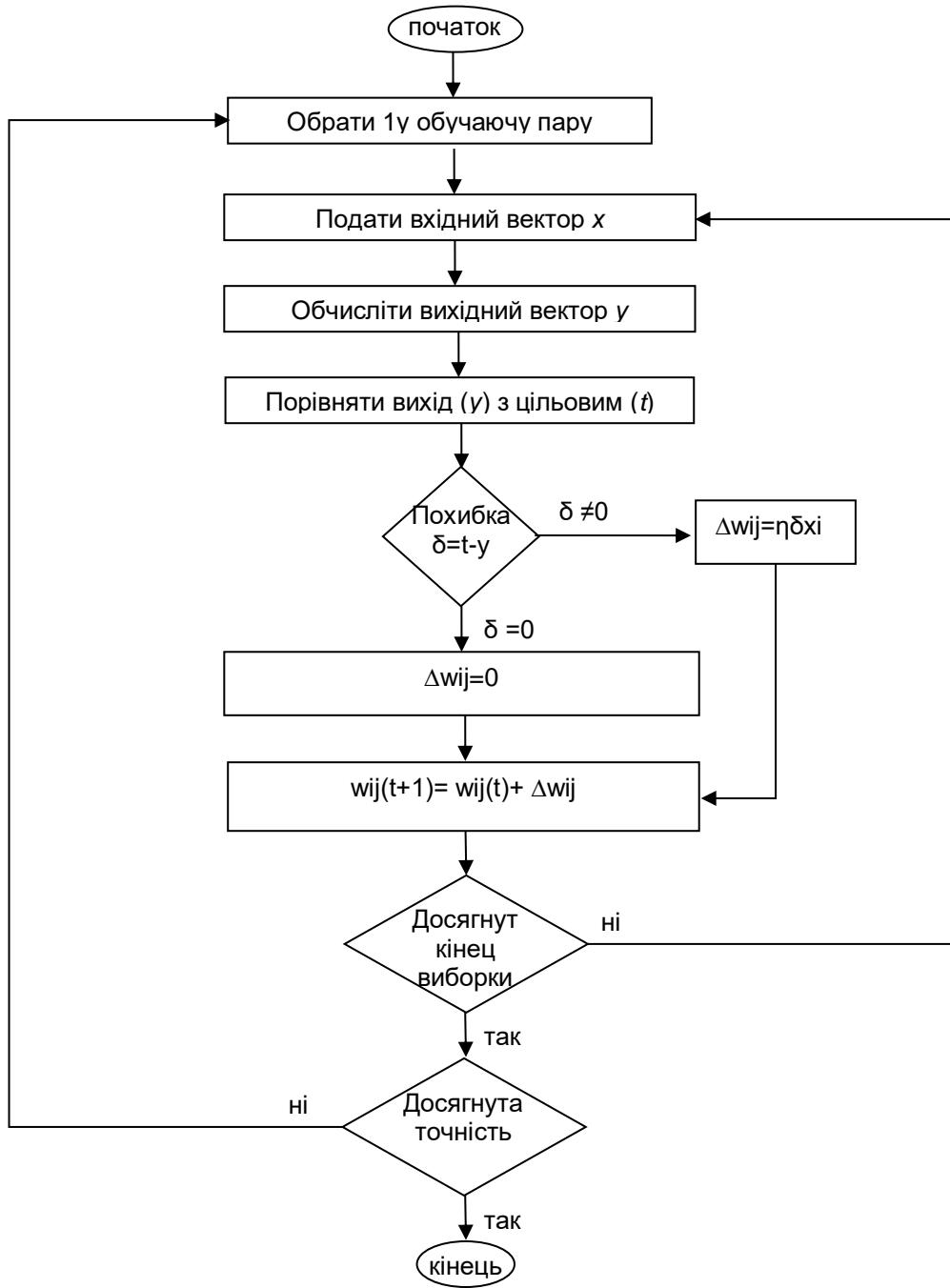
Деякі автори вважають, що один з прихованіх елементів завжди повинен бути лінійним. Це пов'язано з тим, що більшість проблем мають лінійні компоненти, що дуже важливо промоделювати складанням декількох зважених нелінійних функцій. Однак використання лінійної функції активації на двох шарах поспіль недоцільно, тому що це еквівалентно одному лінійному шару з матрицею ваг, що дорівнює добутку матриць ваг 2-х шарів.

**Навчання.** Мережа навчається, щоб для деякої множини входів давати бажану множину виходів. Кожна така вхідна (або вихідна) множина розглядається як вектор. Навчання здійснюється шляхом послідовного подання вхідних векторів з одночасним налаштуванням ваг відповідно до певної процедури.

Розрізняють алгоритми навчання з вчителем і без вчителя. Незважаючи на численні прикладні досягнення, навчання з учителем критикувалося за свою біологічну неправдоподібність.

*Навчання з учителем* передбачає, що для кожного вхідного вектора існує цільовий вектор, що являє собою необхідний вихід. Разом вони називаються навчальною парою. Зазвичай мережа навчається на певній кількості навчальних пар. Вектори навчальної множини послідовно надаються на вхід мережі, обчислюються помилки і ваги підлаштовуються для кожного вектора доти доки помилка по всьому навчальному масиву не досягне прийнятного рівня.

*Навчання з учителем* є набагато більш правдоподібною моделлю навчання в біологічній системі, вона не потребує цільового вектора для виходів і, отже, не вимагає порівняння із зумовленими ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб надання досить близьких вхідних векторів давало однакові виходи.



Після закiнчення навчання мережа готова вирiшити завдання того типу, що її навчили. Рiшення про припинення навчання ухвалюється в результатi аналiзу таких можливих ситуацiй:

- 1) обидвi помилки (i навчання, i тестування) малi. Це означає, що навчання пройшло успiшно, i НМ готова до роботи;

2) помилка тренування мала, а помилка тестування велика. Це означає, що мережа «перенавчилася», вона містить занадто багато вагових коефіцієнтів. У цьому разі архітектуру мережі слід спрощувати шляхом видалення шарів або скорочення кількості нейронів в них;

3) і помилка тренування, і помилка тестування великі. Це означає, що мережа «недонавчена», вона містить занадто мало вагових коефіцієнтів. У цьому разі архітектуру мережі слід ускладнювати. Однак додавання ваг – не панацея; якщо є гіпотеза, що вагових коефіцієнтів досить, необхідно проаналізувати інші можливі причини помилок (наприклад, недостатня кількість навчальних даних).

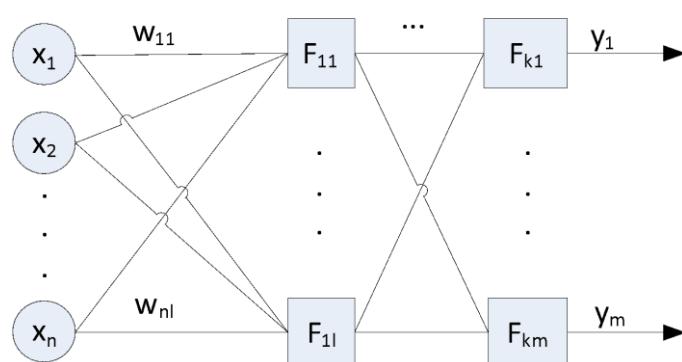
### **Навчальний алгоритм зворотного поширення помилки**

Завдання навчання багатошарових штучних нейронних мереж вирішує алгоритм Румельхарта-Хінтона-Вільямса (алгоритм зворотного поширення помилки). Він був запропонований в різних варіаціях в декількох наукових роботах, існує також безліч поліпшених версій алгоритму.

Нехай задана багатошарова мережа з гладкою функцією активації:

$y = \frac{1}{1 + e^{-s}}$ . Ця функція дуже зручна, тому що її похідна виражається через саму функцію -  $y' = y(1 - y)$ .

Розглянемо схему двошарової мережі. Вхідний шар вважається нульовим шаром. Кожен нейрон наступних шарів видає сигнал  $y$ , перетворюючи зважену суму виходів нейронів попереднього шару.



Навчання мережі вимагає виконання наступних операцій:

1. Вибрати чергову навчальну пару; подати вхідний вектор на вхід мережі.
2. Обчислити вихід мережі.
3. Обчислити різницю між виходом мережі та цільовим вектором навчальної пари.
4. Відкоригувати ваги мережі так, щоб мінімізувати помилку.
5. Повторювати кроки з 1 по 4 для кожного вектору навчальної множини доки помилка на всій множині не досягне прийнятного рівня.

Операції, що виконуються кроками 1 та 2, подібні до тих, які виконуються при функціонуванні вже навченої мережі.

На кроці 3 кожен з виходів мережі вираховується з відповідною компоненти цільового вектору, щоб отримати помилку. Ця помилка використовується на кроці 4 для корекції ваг мережі, причому знак і розмір змін ваг визначаються алгоритмом навчання.

Після достатнього числа повторень цих чотирьох кроків різниця між дійсними виходами і цільовими виходами повинна зменшитися до прийнятної величини.

Кроки 1 і 2 називають «прохід вперед», так як сигнал поширюється по мережі від входу до виходу. Кроки 3, 4 складають «зворотний прохід», тут сигнал помилки поширюється назад по мережі і використовується для підстроювання ваг.

Процедура зворотного поширення може бути застосована до мереж з будь-яким числом шарів. Однак для того, щоб продемонструвати алгоритм, досить двох шарів.

*Прохід вперед.* Кроки 1 і 2 можуть бути виражені в векторній формі наступним чином: подається вхідний вектор  $X$ , на виході отримуємо вектор  $Y$ . Векторна пара вхід-мета  $X$  і  $T$  береться з навчальної множини. Обчислення проводяться над вектором  $X$ , щоб отримати вихідний вектор  $Y$ .

Обчислення в багатошарових мережах виконуються шар за шаром, починаючи з найближчого до входу шару.

*Зворотний прохід.* Підстроювання ваг вихідного шару. Так як для кожного нейрона вихідного шару задано цільове значення, то підстроювання ваг легко здійснюється з використанням модифікованого дельта-правила.

На рис. показаний процес навчання для однієї ваги від нейрона  $p$  в прихованому шарі  $j$  до нейрона  $q$  у вихідному шарі  $k$ . Вихід нейрона шару  $k$ , віднімаючи з цільового значення  $t$ , дає сигнал помилки. Він множиться на похідну функції  $[y (1 - y)]$ , обчислена для цього нейрона шару  $k$ , даючи, таким чином, значення  $\delta$ .

$$\delta_q^k = y_q^k (1 - y_q^k)(t_q - y_q^k)$$

Потім  $\delta_q^k$  множиться на величину  $y_p^j$  нейрона  $j$ , з якого виходить розглянута вага. Цей твір у свою чергу множиться на коефіцієнт швидкості навчання  $\eta$  (зазвичай від 0,01 до 1,0), і результат додається до ваги. Така ж процедура виконується дляожної ваги від нейрона прихованого шару до нейрона у вихідному шарі.

Наступні рівняння ілюструють ці обчислення:

$$\Delta w_{pq,k} = \eta \delta_q^k y_p^j$$

$$w_{pq,k}(n+1) = w_{pq,k}(n) + \Delta w_{pq,k}$$

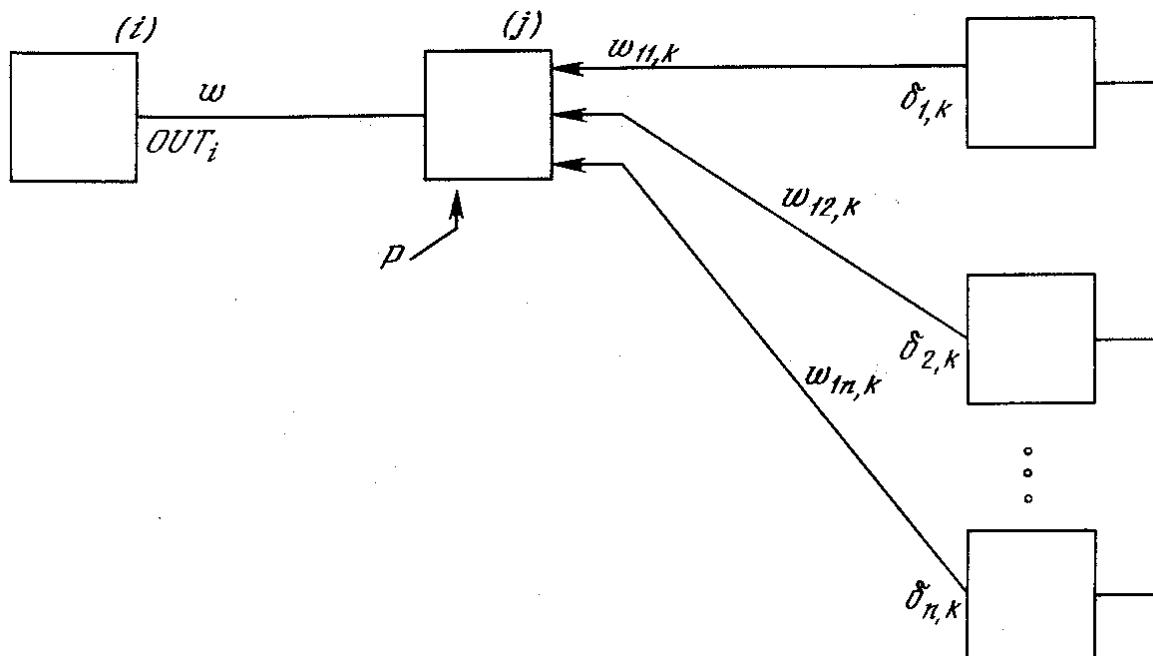
де  $w_{pq,k}(n)$  – вага від нейрона  $p$  в прихованому шарі до нейрона  $q$  у вихідному шарі на кроці  $n$  (до корекції); відзначимо, що індекс  $k$  відноситься до шару, в якому закінчується дана вага;  $w_{pq,k}(n+1)$  – вага на кроці  $n + 1$  (після корекції);  $\delta_{q,k}$  -  $\delta$  для нейрона  $q$ , в вихідному шарі  $k$ ;  $y_{p,j}$  - вихід для нейрона  $p$  у прихованому шарі  $j$ .

*Підстроювання ваг прихованого шару.* Розглянемо один нейрон в прихованому шарі, що передує вихідному шару. При проході вперед цей нейрон передає свій вихідний сигнал нейронам в вихідному шарі через ваги, що їх з'єднують. Під час навчання ці ваги функціонують в зворотному

порядку, пропускаючи  $\delta$  від вихідного шару назад до прихованого шару. Кожна з цих ваг множиться на величину  $\delta$  нейрона, до якого вона приєднана у вихідному шарі.  $\delta$ , необхідна для нейрона прихованого шару, виходить підсумуванням всіх таких творів і множенням на похідну стискаючої функції:

$$\delta_{q,k} = y_{p,j}(1 - y_{p,j}) \left[ \sum_q \delta_{q,k} w_{pq,k} \right]$$

Коли значення  $\delta$  отримано, ваги першого прихованого шару можуть бути скоректовані, відповідно до описаних вище співвідношень.



Для кожного нейрона в даному прихованому шарі має бути обчислено  $\delta$  і налаштовані всі ваги, асоційовані з цим шаром. Цей процес повторюється шар за шаром у напрямку до входу, поки все ваги не будуть скоректовані.

### **Проблеми навчання нейронних мереж**

Незважаючи на численні успішні застосування зворотного поширення, воно має ряд недоліків. Найбільше неприємностей приносить невизначене довгий процес навчання. Тривалий час навчання може бути результатом неоптимального вибору довжини кроку. Невдачі в навчанні

зазвичай виникають з двох причин: паралічу мережі і попадання в локальний мінімум.

*Параліч мережі.* У процесі навчання мережі значення ваг можуть в результаті корекції стати дуже великими. Це може привести до того, що всі або більшість нейронів будуть функціонувати при дуже великих значеннях  $y$ , в області, де похідна дуже мала. Так як помилка пропорційна похідній, то процес навчання може практично завмерти. У теоретичному відношенні ця проблема погано вивчена. Зазвичай цього уникають зменшенням розміру кроку  $\eta$ , але це збільшує час навчання. Різні евристики використовувалися для запобігання паралічу або для відновлення після нього.

*Локальні мінімуми.* Зворотне поширення використовує різновид градієнтного спуску, тобто здійснює спуск по поверхні помилки, безперервно підлаштовуючи ваги в напрямку до мінімуму. Поверхня помилки складної мережі складна, мережа може потрапити в локальний мінімум (неглибоку долину), коли поруч є набагато більш глибокий мінімум. У точці локального мінімуму всі напрямки ведуть вгору, і мережа не здатна з нього відібратися.

*Розмір кроку.* Якщо розмір кроку дуже малий, то збіжність надто повільна, якщо ж дуже великий, то може виникнути параліч або нестійкість.

*Нестійкість.* Процес навчання повинен бути таким, щоб мережа навчалася на всій навчальній множині без пропусків того, що вже вивчено. Алгоритм вимагає, щоб всі навчальні образи пред'являлися перед кожною корекцією параметрів. Це випливає з необхідності підсумовувати функцію помилки за всіма даними з навчальної множини. У цьому випадку алгоритм завжди сходиться, хоча кількість ітерацій може виявитися занадто великою. Якщо корекції параметрів проводяться після пред'явлення кожного образу,

або навіть після розрахунку кожної компоненти градієнта, то алгоритм може не зйтися взагалі, якщо:

- а) образи пред'являються не в довільному порядку;
- б) навчальна множина постійно змінюється, і кожен образ пред'являється малу кількість разів; це зустрічається для систем, що працюють в реальному часі.

Проблема а) вирішується випадковим пред'явленням образів або застосуванням алгоритму навчання з пакетною корекцією. Проблема б) від початку властива методу зворотного поширення і усувається вибором іншого алгоритму навчання або іншої моделі нейронної мережі.

## Тема 18. НМ в задачах апроксимації та прогнозування

### НМ в задачах апроксимації

У задачах прогнозування та апроксимації важлива точність, тому використовуються архітектури нейронних мереж, що розширяються, тобто такі, в яких кількість нейронів на першому шарі більше, ніж на вхідному.

Для оцінки якості НМ в задачах апроксимації та прогнозування використовуються такі функції втрат:

- a. mse – середня квадратична помилка;
- b. rmse – корінь з середньоквадратичної помилки;
- c. mae – середня абсолютна помилка;
- d. mape – середня помилка у відсотках;
- e. msle – середня квадратично-логарифмічна помилка.

Параметри налаштування нейронної мережі **nnet** з одним прихованим шаром:

x	Матриця навчальної вибірки.
y	Матриця відповідей для навчальної вибірки (повинна бути перетворена з використанням <code>class.ind</code> ).
weights	Ваги об'єктів, що беруться за одиниці в разі пропуску.
size	Кількість нейронів прихованого шару.
formula	Формула вигляду <code>class ~ x1 + x2 + ...</code>
data	Data frame – вибірка.
subset	Вектор індексів об'єктів, на яких слід провести навчання.
na.action	Параметр, що визначає, що робити з НА-значеннями. За замовчуванням – закінчення процедури.
Wts	Вектор початкових значень параметрів. У разі пропуску задається випадково.
mask	Логічний вектор, що визначає, які з параметрів необхідно оптимізувати. За замовчуванням всі.
linout	Перемикання на linear output units. За замовчуванням logistic output units.
entropy	Перемикач для ентропії.
softmax	Перемикач для softmax (log-linear model) і maximum conditional likelihood fitting. linout, entropy, softmax і censored виключають одне одного.
censored	Варіант softmax, в якому ненульові мітки означають можливість приналежності до цього класу.

	Отже, для softmax вектор $(0, 1, 1)$ означає, що об'єкт належить до обох класів 2 і 3, але для censored він означає, що об'єкт належить або до класу 2, або до класу 3.
skip	Перемикання додавання зв'язків між входом і виходом.
rang	Випадкові початкові значення ваг беруться з діапазону $[-rang, rang]$ .
decay	Параметр для weight decay (скорочення ваг). За замовчуванням 0.
maxit	Кількість максимальної кількості ітерацій. За замовчуванням 100.
trace	Містить tracing optimization. За замовчуванням TRUE.
MaxNWts	Максимально допустима кількість ваг.
abstol	Зупинка, якщо значення критерію під час налаштування опускається нижче abstol.
reldtol	Зупинка, якщо оптимізатор не в змозі зменшити критерій налаштування принаймні на $1 - reldtol$ .

## Лабораторна робота 14

### NEURAL NETWORKS FOR APPROXIMATION

#### Download the data and libraries

```

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union

library(nnet)
library(ggplot2)
library(knitr)
library (psych)

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
## 
##     %+%, alpha

#Set Working Directory
#setwd('D:/ML')
#OR Choose your Directory in 'Files' and click on 'More' -> 'Set as Working Directory' #Download file to the table. Source file is 'flats.csv'
f <- read.csv2('flats.csv', header = TRUE, encoding = 'UNICOD')
describe(f)

##          vars   n    mean      sd median trimmed      mad min      max
## rooms       1 216    2.01    0.97      2    1.94    1.48    1      6
## location*   2 217    1.27    0.44      1    1.21    0.00    1      2
## condition*  3 217    1.77    0.42      2    1.84    0.00    1      2
## m2          4 217   76.33   38.02     67    70.94   28.17   21    280
## type*       5 216    1.20    0.40      1    1.13    0.00    1      2
## price        6 217 82427.45 82183.66   59548 67365.84 35609.09    1 750000
##                  range skew kurtosis      se
## rooms            5  0.73    0.44    0.07
## location*       1  1.04   -0.91    0.03
## condition*      1 -1.30   -0.30    0.03
## m2              259  1.77    4.61    2.58
## type*           1  1.46    0.14    0.03
## price          749999  4.58   29.38  5578.99

```

**Висновок:** кількість спостережень – 217, кількість змінних – 6. Аналіз основних показників описової статистики за кожною змінною показав, що є пропущені значення в змінних – rooms, type. Змінні m2 та price мають викиди.

### Missing data, fill n/a with average

```
f_fill <- f
##with average for integer vars
f_fill$rooms <- ifelse(is.na(f$rooms), round(mean(f$rooms, na.rm = TRUE)), f$rooms)
##the most frequent for categorical vars
f_fill$type <- ifelse(is.na(f$type), which.max(table(f$type)), f$type)
f_fill$type <- as.factor(f_fill$type)
f <- f_fill
```

**Висновок:** для заповнення пропусків обраний варіант заповнення середніми для кількісних змінних і найбільш частотними для якісних змінних.

### Ejections (outside the three sigma).

```
f_ej <- f
f_ej$price <- ifelse(f$price < mean(f$price)+sd(f$price)*3, f$price, mean(f$price)+sd(f$price)*3)
describe(f_ej$price)

##      vars     n    mean        sd median   trimmed      mad    min     max
## X1      1 217 78680.61 59689.36 59548 67365.84 35609.09    1 328978.4
##          range skew kurtosis      se
## X1 328977.4 2.02     4.42 4051.98

f <- f_ej
```

**Висновок:** для корекції викидів змінної price обраний варіант заповнення граничними значеннями.

### Factors as numeric

```
f$location <- as.numeric(f$location)
f$condition <- as.numeric(f$condition)
f$type <- as.numeric(f$type)
```

**Висновок:** якісні змінні перетворено на кількісні.

### Features Scaling

```
f_sc <- f
f_sc$price <- scale(f$price)
f_sc$rooms <- scale(f$rooms)
f_sc$m2 <- scale(f$m2)
head (f_sc)

##           rooms location condition          m2  type     price
## 1 -0.01426214         2           1 -0.6925778    2 -0.7317990
## 2 -1.04589039         1           1 -1.0345033    2 -0.7317990
## 3  1.01736611         2           1 -0.2454445    2 -0.2291968
## 4 -0.01426214         2           1 -1.4553347    2 -1.0668671
```

```
## 5 -1.04589039      2      1  0.1490848      1 -0.3129638
## 6  1.01736611      1      1  0.1490848      2  0.1058713
```

**Висновок:** алгоритми нейронних мереж потребують шкалювання, виконано шкалювання кількісних змінних.

### Splitting the scaled dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f_sc$price, SplitRatio = 0.8)
f_train = subset(f_sc, split == TRUE)
f_test = subset(f_sc, split == FALSE)
```

**Висновок:** датасет розподілений на навчальну та тестову вибірки.

### Fitting the NN

```
set.seed(333)
ff_ap <- nnet(data = f_train, price ~ m2 + location + type, linout = TRUE
, size = 6, maxit = 10000)
library(graphics)
source(file = 'plot.nnet.R')
plot.nnet(ff_ap)

## Loading required package: scales

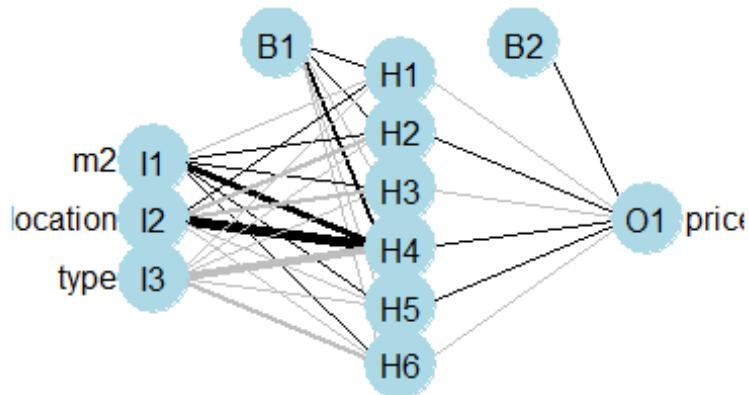
##
## Attaching package: 'scales'

## The following objects are masked from 'package:psych':
##
##     alpha, rescale

## Loading required package: reshape

##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename
```



Висновок: на основі змінних m2, location, type побудовано двошарову нейронну мережу для прогнозування цін на нерухомість.

### Prediction

```
p_ff_ap <- predict(ff_ap, f_test)

train_mse_ff_ap <- sum((f_train$price-predict(ff_ap, f_train))^2)/length(f_train$price)
test_mse_ff_ap <- sum((f_test$price-p_ff_ap)^2)/length(p_ff_ap)

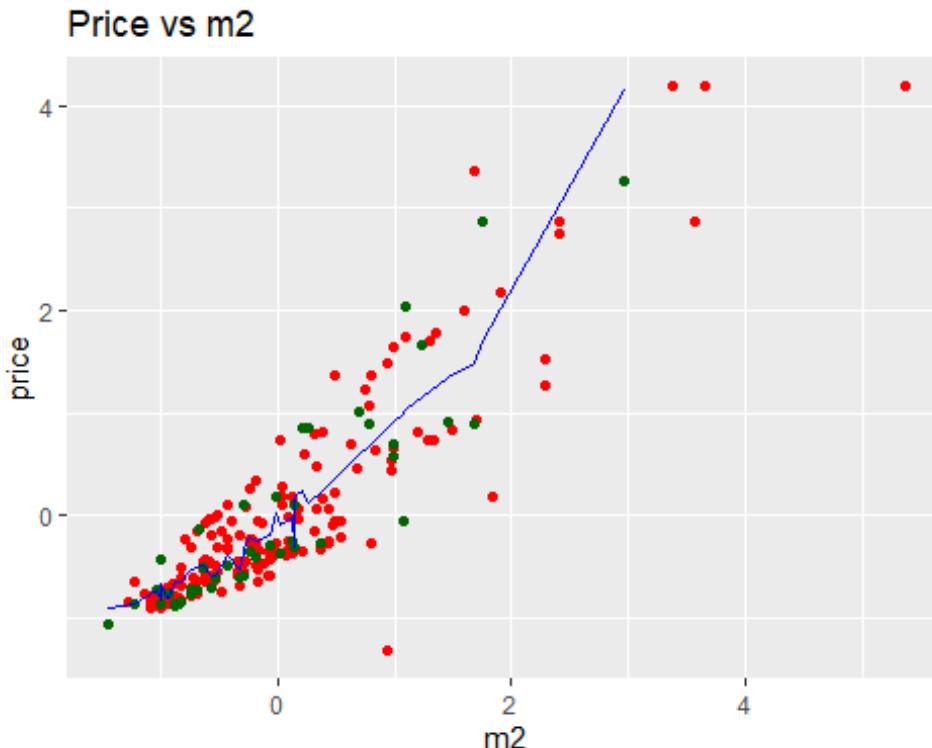
train_mse_ff_ap
## [1] 0.1546619

test_mse_ff_ap
## [1] 0.1778553
```

Висновок: значення середньоквадратичної помилки на навчальній вибірці – 0,155, на тестовій вибірці – 0,178.

### Visualising

```
library(ggplot2)
ggplot() +
  geom_point(aes(f_train$m2, f_train$price), colour = 'red') +
  geom_point(aes(f_test$m2, f_test$price), colour = 'dark green') +
  geom_line(aes(f_test$m2, p_ff_ap), colour = 'blue') +
  ggtitle('Price vs m2') +
  xlab('m2') +
  ylab('price')
```



**Висновок:** на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Fit NN-2

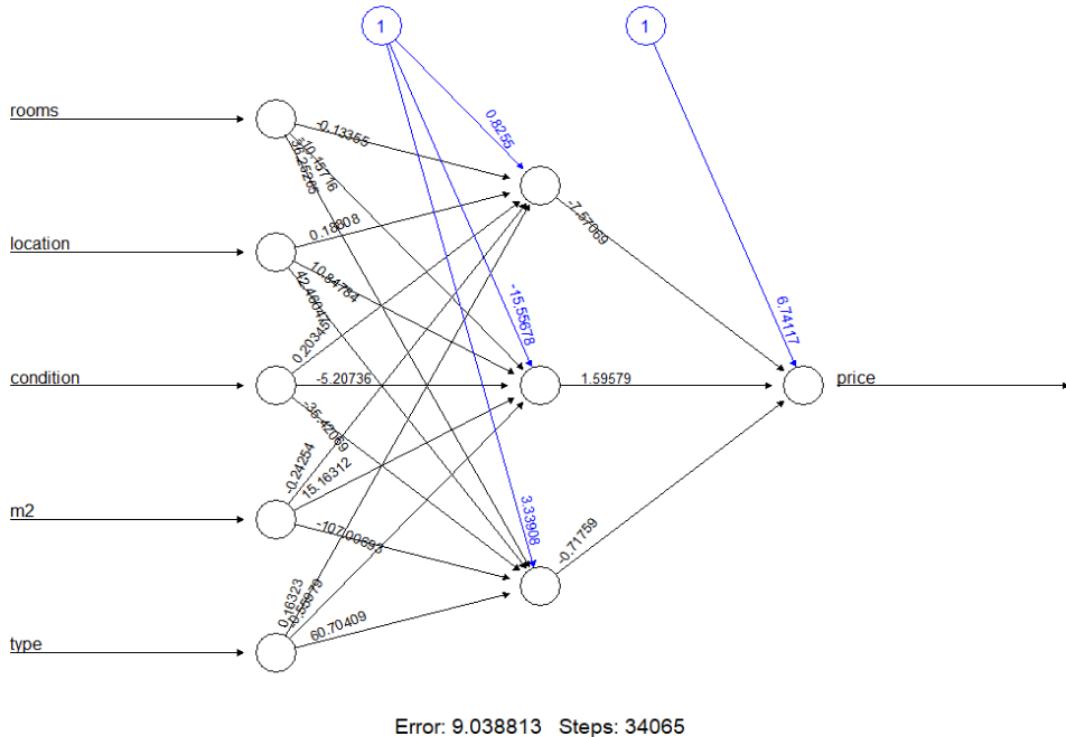
```
library(neuralnet)

##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':
##      compute

# fit neural network
set.seed(2)
nn = neuralnet(price ~ ., f_train, hidden = 3 , linear.output = T)

# plot neural network
plot(nn)
```



**Висновок:** на основі усіх змінних побудовано двошарову нейронну мережу для прогнозування цін на нерухомість.

### Prediction

```
p_nn <- predict(nn, f_test)

train_mse_nn <- sum((f_train$price-predict(nn, f_train))^2)/length(f_train$price)
test_mse_nn <- sum((f_test$price-p_nn)^2)/length(p_nn)

train_mse_nn
## [1] 0.104495

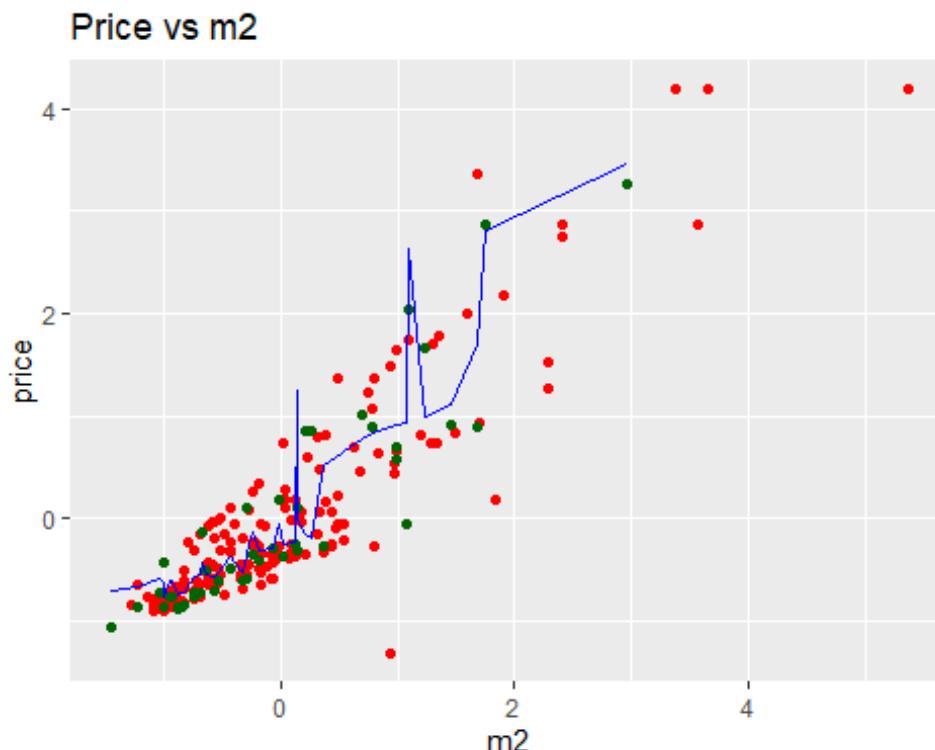
test_mse_nn
## [1] 0.2036023
```

**Висновок:** значення середньоквадратичної помилки на навчальній вибірці зменшилося – 0,104, на тестовій вибірці зросло – 0,204, що свідчить про перенавчання моделі.

### Visualising

```
library(ggplot2)
ggplot() +
  geom_point(aes(f_train$m2, f_train$price), colour = 'red') +
  geom_point(aes(f_test$m2, f_test$price), colour = 'dark green') +
  geom_line(aes(f_test$m2, p_nn), colour = 'blue') +
  ggtitle('Price vs m2') +
```

```
xlab('m2') +
ylab('price')
```



**Висновок:** на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Довідка: neuralnet

```
neuralnet(formula, data, hidden = 1, threshold = 0.01,
stepmax = 1e+05, rep = 1, startweights = NULL,
learningrate.limit = NULL, learningrate.factor = list(minus = 0.5,
plus = 1.2), learningrate = NULL, lifesign = "none",
lifesign.step = 1000, algorithm = "rprop+", err.fct = "sse",
act.fct = "logistic", linear.output = TRUE, exclude = NULL,
constant.weights = NULL, likelihood = FALSE)
```

### Arguments

formula	a symbolic description of the model to be fitted.
data	a data frame containing the variables specified in formula.
hidden	a vector of integers specifying the number of hidden neurons (vertices) in each layer.
threshold	a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria.
stepmax	the maximum steps for the training of the neural network. Reaching this maximum leads to a stop of the neural network's training process.

Rep	the number of repetitions for the neural network's training.
startweights	a vector containing starting values for the weights. Set to NULL for random initialization.
learningrate.limit	a vector or a list containing the lowest and highest limit for the learning rate. Used only for RPROP and GRPROP.
learningrate.factor	a vector or a list containing the multiplication factors for the upper and lower learning rate. Used only for RPROP and GRPROP.
learningrate	a numeric value specifying the learning rate used by traditional backpropagation. Used only for traditional backpropagation.
lifesign	a string specifying how much the function will print during the calculation of the neural network. 'none', 'minimal' or 'full'.
lifesign.step	an integer specifying the stepsize to print the minimal threshold in full lifesign mode.
algorithm	a string containing the algorithm type to calculate the neural network. The following types are possible: 'backprop', 'rprop+', 'rprop-', 'sag', or 'slr'. 'backprop' refers to backpropagation, 'rprop+' and 'rprop-' refer to the resilient backpropagation with and without weight backtracking, while 'sag' and 'slr' induce the usage of the modified globally convergent algorithm (grprop). See Details for more information.
err.fct	a differentiable function that is used for the calculation of the error. Alternatively, the strings 'sse' and 'ce' which stand for the sum of squared errors and the cross-entropy can be used.
act.fct	a differentiable function that is used for smoothing the result of the cross product of the covariate or neurons and the weights. Additionally the strings, 'logistic' and 'tanh' are possible for the logistic function and tangent hyperbolicus.
linear.output	logical. If act.fct should not be applied to the output neurons set linear output to TRUE, otherwise to FALSE.
exclude	a vector or a matrix specifying the weights, that are excluded from the calculation. If given as a vector, the exact positions of the weights must be known. A matrix with n-rows and 3 columns will exclude n weights, where the first column stands for the layer, the second column for the input neuron and the third column for the output neuron of the weight.
constant.weights	a vector specifying the values of the weights that are excluded from the training process and treated as fix.
likelihood	logical. If the error function is equal to the negative log-likelihood function, the information criteria AIC and BIC will be calculated. Furthermore the usage of confidence.interval is meaningfull.

neuralnet returns an object of class nn. An object of class nn is a list containing at most the following components:

call	the matched call.
response	extracted from the data argument.
covariate	the variables extracted from the data argument.
model.list	a list containing the covariates and the response variables extracted from the formula argument.
err.fct	the error function.
act.fct	the activation function.
data	the data argument.
net.result	a list containing the overall result of the neural network for every repetition.
weights	a list containing the fitted weights of the neural network for every repetition.
generalized.weights	a list containing the generalized weights of the neural network for every repetition.
result.matrix	a matrix containing the reached threshold, needed steps, error, AIC and BIC (if computed) and weights for every repetition. Each column represents one repetition.
startweights	a list containing the startweights of the neural network for every repetition.

### **НМ в задачах прогнозування**

Застосування повнозв'язних НМ в задачах прогнозування часових рядів має деякі особливості:

- 1) для видалення тренду зазвичай на вхід моделі подається не початковий ряд, а перші різниці;
- 2) оскільки як пояснюючі змінні виступають попередні значення ряду, важливо визначитися, скільки лагів є суттєвими для конкретної задачі;
- 3) оскільки кількість лагів обмежена, довгострокові тенденції в такій моделі не враховуються.

### **Лабораторна робота 15**

## **NEURAL NETWORKS FOR TIME SERIES**

#### **Download the data and libraries**

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union

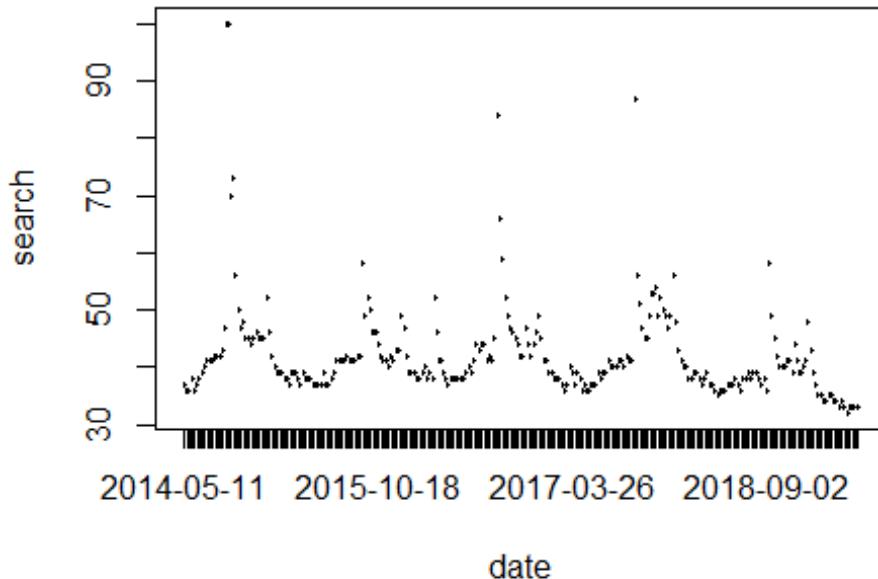
library(nnet)
library(ggplot2)
library(knitr)
library (psych)

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
## 
##     %+%, alpha

#Set Working Directory
#setwd('D:/ML')
#OR Choose your Directory in 'Files' and click on 'More' -> 'Set as
```

```
Working Directory' #Download file to the table.
f <- read.csv('iPhone.csv', header = TRUE, encoding = 'UNICOD')
plot(f)
```



Висновок: завантажено часовий ряд.

## Transforming data

```
#to stationarity
df = diff(f$search, differences = 1)
df <- as.data.frame(df)

#scaling
ymax <- max(df$df)
ymin <- min(df$df)
df$y <- (df$df - ymin) / (ymax - ymin)

#lagging
df$x1 <- dplyr::lag(df$y)
df$x2 <- dplyr::lag(df$y,2)
df$x3 <- dplyr::lag(df$y,3)
head (df)

##   df      y      x1      x2      x3
## 1 -1 0.3571429     NA     NA     NA
## 2  0 0.3690476 0.3571429     NA     NA
## 3  2 0.3928571 0.3690476 0.3571429     NA
## 4 -2 0.3452381 0.3928571 0.3690476 0.3571429
## 5  1 0.3809524 0.3452381 0.3928571 0.3690476
## 6  1 0.3809524 0.3809524 0.3452381 0.3928571
```

**Висновок:** ряд зведений до стаціонарного, виконано шкалювання, додані лагові змінні.

### Splitting the scaled dataset into the TRAIN set and TEST set

```
N = nrow(df)
n = round(N *0.7, digits = 0)
train = df[4:n, ]
test = df[(n+1):N, ]
```

**Висновок:** датасет розділений на навчальну та тестову вибірки, послідовно починаючи з 4 рядка.

### Fitting the NN

```
set.seed(11)
ff_ts <- nnet::nnet(train[,2:4], train$y, linout = TRUE ,size = 10, maxit
= 10000)
library(graphics)
source(file = 'plot.nnet.R')
plot.nnet(ff_ts)

## Loading required package: scales

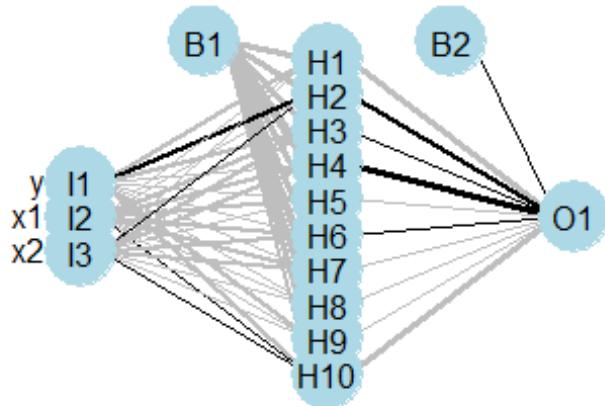
##
## Attaching package: 'scales'

## The following objects are masked from 'package:psych':
##
##     alpha, rescale

## Loading required package: reshape

##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename
```



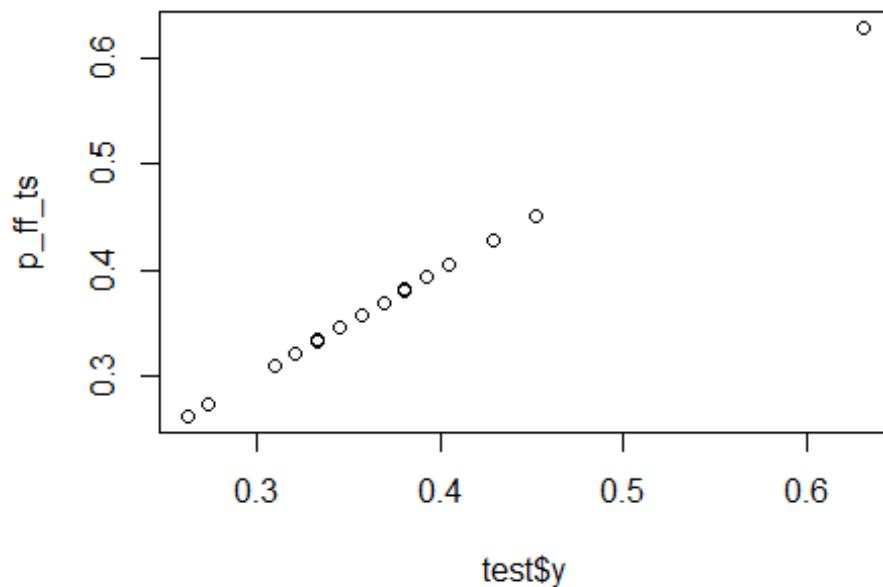
**Висновок:** побудовано двошарову нейронну мережу для прогнозування кількості пошукових запитів.

### Prediction

```
p_ff_ts <- predict(ff_ts, test[,2:4])
mse_ff_ts<-sum((test[,1]-p_ff_ts)^2)/length(p_ff_ts)
mse_ff_ts

## [1] 12.46352

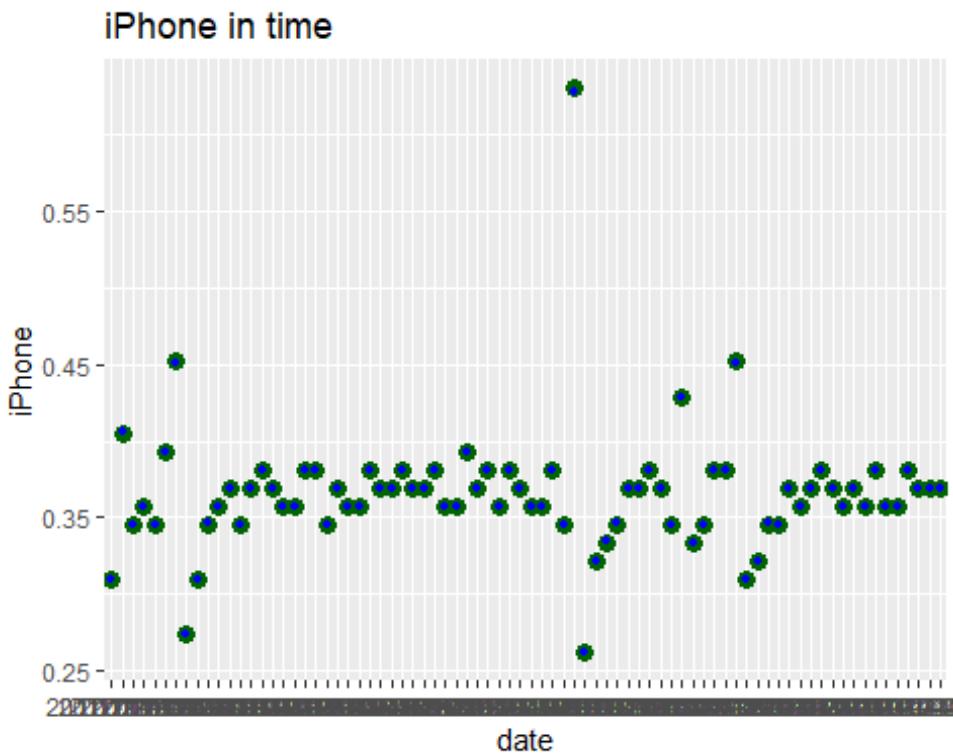
plot(test$y, p_ff_ts)
```



Висновок: значення середньоквадратичної помилки на тестовій вибірці – 12,464.  
Прогнозні значення близькі до лінії ідеальних прогнозів.

## Visualising

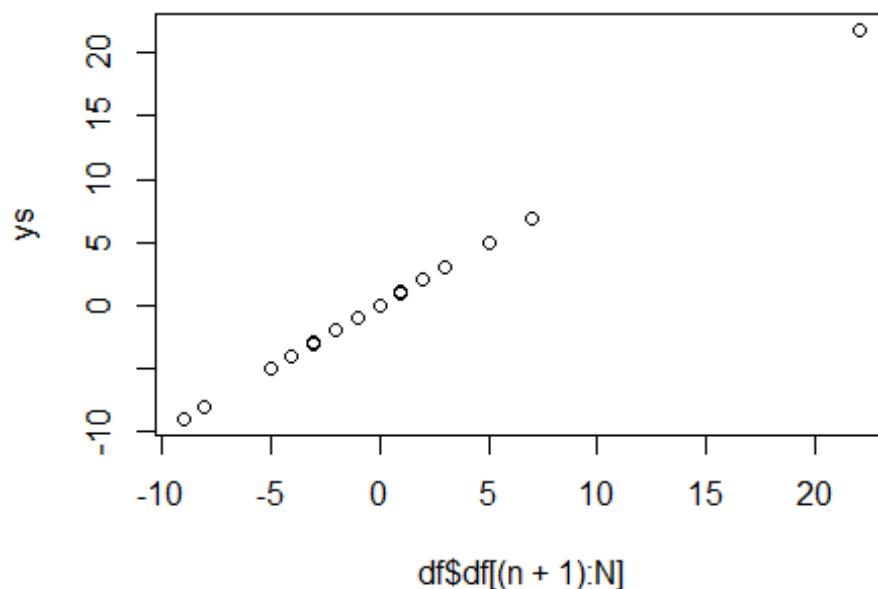
```
library(ggplot2)
ggplot() +
# geom_point(aes(f$date[4:n], train$y), colour = 'red') +
  geom_point(aes(f$date[(n+1):N], test$y), colour = 'dark green', size = 3)
+
  geom_point(aes(f$date[(n+1):N], p_ff_ts), colour = 'blue', size = 1) +
  ggtitle('iPhone in time') +
  xlab('date') +
  ylab('iPhone')
```



Висновок: на графіку зеленим позначені точки тестової вибірки, синім – модельні значення.

### Inverting

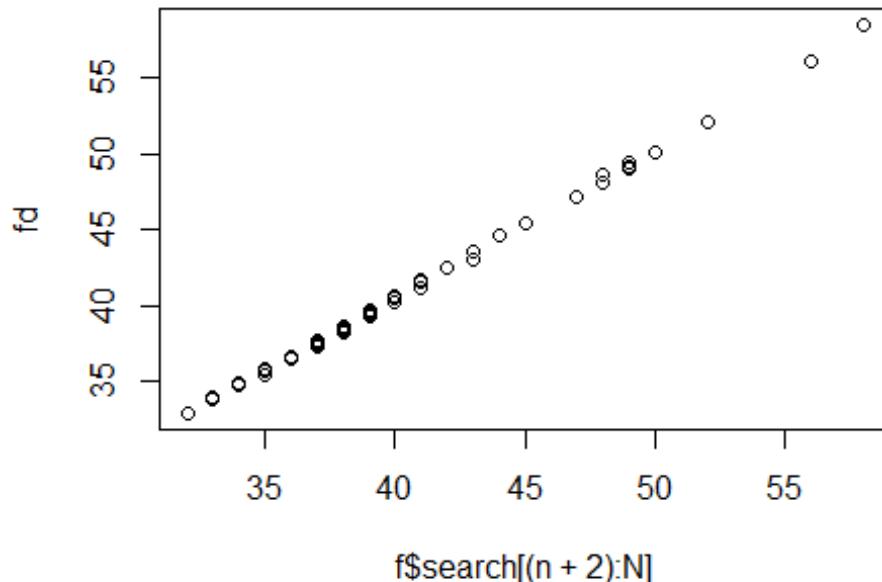
```
# invert scaling
ys = p_ff_ts * (ymax - ymin) + ymin
plot(df$df[(n+1):N],ys)
```



```
# invert differencing
fd = numeric(N-n-1)
fd[1] <- f$search[n+1] + ys[1]

for(i in 2:(N-n-1)){
    fd[i] = fd[i-1] + ys[i]
}

library(ggplot2)
plot(f$search[(n+2):N], fd)
```

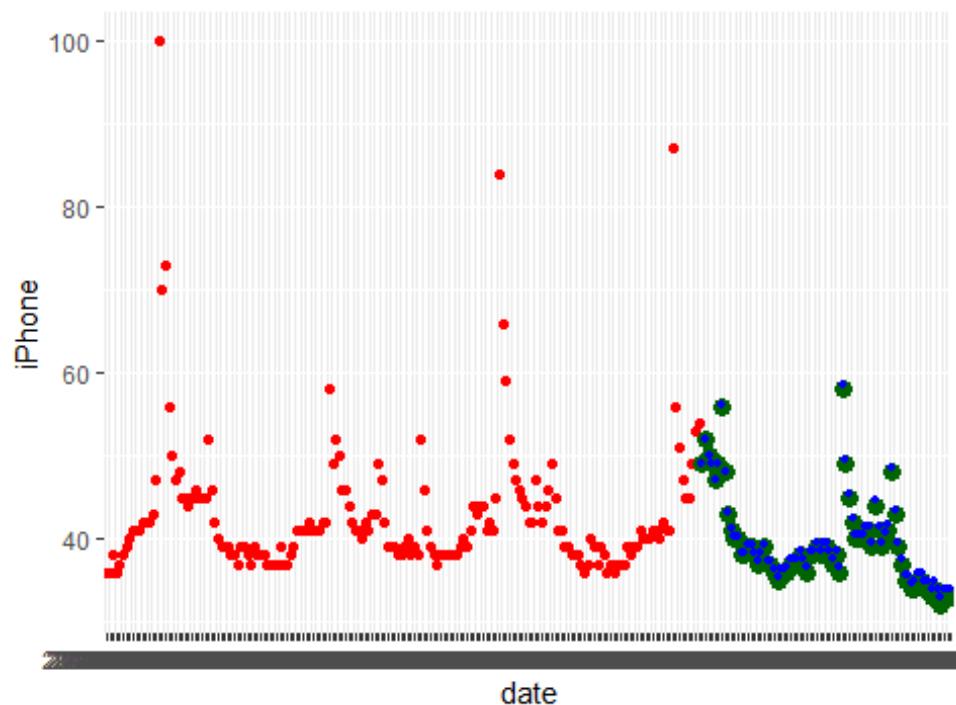


**Висновок:** виконано зворотне перетворення даних. Прогнозні значення близькі до лінії ідеальних прогнозів.

### Visualising

```
library(ggplot2)
ggplot() +
  geom_point(aes(f$date[2:(n+1)], f$search[2:(n+1)]), colour = 'red') +
  geom_point(aes(f$date[(n+2):N], f$search[(n+2):N]), colour = 'dark green',
             size = 3) +
  geom_point(aes(f$date[(n+2):N], fd), colour = 'blue', size = 1) +
  ggtitle('iPhone in time') +
  xlab('date') +
  ylab('iPhone')
```

iPhone in time



Висновок: на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Тема 19. НМ в задачах класифікації

У задачах класифікації важлива узагальнююча здатність мережі, тому використовуються архітектури, що зважуються, тобто такі, в яких кількість нейронів на першому шарі менше, ніж у вхідному.

Для оцінки якості НМ в задачах класифікації використовуються такі функції витрат:

- a) binary accuracy – середня точність за всіма прогнозами в задачах бінарної класифікації;
- b) categorical accuracy – середня точність за всіма прогнозами в задачах багатокласової класифікації.

## Лабораторна робота 16

### NEURAL NETWORKS FOR CLASSIFICATION

#### Download the data and libraries

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##     filter, lag

## The following objects are masked from 'package:base':
##     intersect, setdiff, setequal, union

library(nnet)
library(ggplot2)
library(knitr)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
library (psych)

##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##     %+%, alpha
```

```
describe(f)
```

	vars	n	mean	sd	median	trimmed	mad	min
## id*	1	600	300.50	173.35	300.5	300.50	222.39	1.00
## age	2	600	42.40	14.42	42.0	42.34	17.79	18.00
## sex*	3	600	1.50	0.50	1.5	1.50	0.74	1.00
## region*	4	600	2.23	1.29	2.0	2.16	1.48	1.00
## income	5	600	27524.03	12899.47	24925.3	26432.88	12854.96	5014.21
## married*	6	600	1.66	0.47	2.0	1.70	0.00	1.00
## children	7	600	1.01	1.06	1.0	0.89	1.48	0.00
## car*	8	600	1.49	0.50	1.0	1.49	0.00	1.00
## mortgage*	9	600	1.35	0.48	1.0	1.31	0.00	1.00
## delays*	10	600	1.51	0.50	2.0	1.51	0.00	1.00
			max	range	skew	kurtosis	se	
## id*		600.0	599.00	0.00	-1.21	7.08		
## age		67.0	49.00	0.04	-1.17	0.59		
## sex*		2.0	1.00	0.00	-2.00	0.02		
## region*		4.0	3.00	0.38	-1.58	0.05		
## income		63130.1	58115.89	0.66	-0.33	526.62		
## married*		2.0	1.00	-0.67	-1.55	0.02		
## children		3.0	3.00	0.55	-1.04	0.04		
## car*		2.0	1.00	0.03	-2.00	0.02		
## mortgage*		2.0	1.00	0.64	-1.60	0.02		
## delays*		2.0	1.00	-0.03	-2.00	0.02		

Висновок: для побудови моделі банківського скрингу використані дані про наявність прострочених платежів по кредиту. Кількість спостережень – 600, кількість змінних – 10, з них якісних – 7, кількісних – 3. Пропущених значень і викидів немає.

## Features Scaling

```
f <- f[-1]
f$age = scale(f$age)
f$children = scale(f$children)
f$income = scale(f$income)
head(f)

##      age   sex   region   income married   children car mortgage
## 1  0.3885629 FEMALE INNER_CITY -0.7735227  NO -0.01104012 NO      NO
## 2 -0.1660318 MALE    TOWN  0.1985406 YES  1.88155126 YES     YES
## 3  0.5965360 FEMALE INNER_CITY -0.8487661 YES -0.95733581 YES      NO
## 4 -1.3445456 FEMALE    TOWN -0.5541803 YES  1.88155126 NO      NO
## 5  1.0124820 FEMALE   RURAL  1.7870712 YES -0.95733581 NO      NO
## 6  1.0124820 FEMALE    TOWN  0.8020151 YES  0.93525557 NO      NO
##   delays
## 1   YES
## 2   NO
## 3   YES
## 4   YES
## 5   NO
## 6   NO
```

**Висновок:** моделі класифікації вимагають попереднього шкалювання кількісних змінних.

### Splitting the scaled dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f$income, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

**Висновок:** датасет розподілено на навчальну та тестову вибірки.

### Fitting the NN

```
set.seed(11)
ff_cl <- nnet(data = f_train, delays ~ married + children + income + age,
size = 3, maxit = 1000)
library(graphics)
source(file = 'plot.nnet.R')
plot.nnet(ff_cl)

## Loading required package: scales

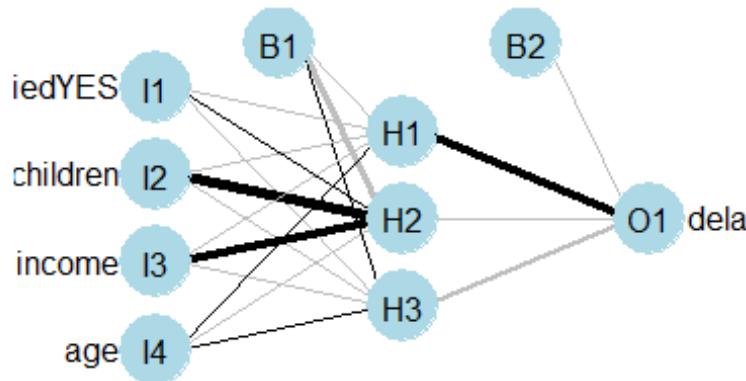
##
## Attaching package: 'scales'

## The following objects are masked from 'package:psych':
##
##     alpha, rescale

## Loading required package: reshape

##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename
```



**Висновок:** на основі змінних `married`, `children`, `income`, `age` побудовано двошарову нейронну мережу для прогнозування класу позичальника.

## Predicting

```
p_ff_c1 <- predict(ff_c1, f_test, type = "class")
p <- as.factor(p_ff_c1)
```

**Висновок:** визначені класи об'єктів (вектор p).

## Confusion Matrix

```
cm = table(f_test[, 'delays'], p)
print(cm)

##          p
##      NO YES
##  NO  89  12
##  YES 12  87
```

**Висновок:** точність моделі –  $(89+87) / 200 = 88\%$ , частка невірно класифікованих випадків –  $(12+12) / 200 = 12\%$ . Чутливість моделі –  $87 / (12+87) = 88\%$ , специфічність –  $89 / (89+12) = 88\%$ , тобто модель однаково чутлива до виявлення позитивних та негативних випадків.

## Довідка: nnet

```
nnet(x, ...)

## S3 method for class 'formula'
nnet(formula, data, weights, ...,
     subset, na.action, contrasts = NULL)
```

```
## Default S3 method:  
nnet(x, y, weights, size, Wts, mask,  
    linout = FALSE, entropy = FALSE, softmax = FALSE,  
    censored = FALSE, skip = FALSE, rang = 0.7, decay = 0,  
    maxit = 100, Hess = FALSE, trace = TRUE, MaxNWts = 1000,  
    abstol = 1.0e-4, reltol = 1.0e-8, ...)
```

## Arguments

formula	A formula of the form class ~ x1 + x2 + ...
x	matrix or data frame of x values for examples.
y	matrix or data frame of target values for examples.
weights	(case) weights for each example – if missing defaults to 1.
size	number of units in the hidden layer. Can be zero if there are skip-layer units.
data	Data frame from which variables specified in formula are preferentially to be taken.
subset	An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named).
na.action	A function to specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named).
contrasts	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
Wts	initial parameter vector. If missing chosen at random.
mask	logical vector indicating which parameters should be optimized (default all).
linout	switch for linear output units. Default logistic output units.
entropy	switch for entropy (= maximum conditional likelihood) fitting. Default by least-squares.
softmax	switch for softmax (log-linear model) and maximum conditional likelihood fitting. linout, entropy, softmax and censored are mutually exclusive.
censored	A variant on softmax, in which non-zero targets mean possible classes. Thus for softmax a row of (0, 1, 1) means one example each of classes 2 and 3, but for censored it means one example whose class is only known to be 2 or 3.
skip	switch to add skip-layer connections from input to output.
rang	Initial random weights on [-rang, rang]. Value about 0,5 unless the inputs are large, in which case it should be chosen so that rang * max( x ) is about 1.
decay	parameter for weight decay. Default 0.
maxit	maximum number of iterations. Default 100.

Hess	If true, the Hessian of the measure of fit at the best set of weights found is returned as component Hessian.
trace	switch for tracing optimization. Default TRUE.
MaxNWts	The maximum allowable number of weights. There is no intrinsic limit in the code, but increasing MaxNWts will probably allow fits that are very slow and time-consuming.
abstol	Stop if the fit criterion falls below abstol, indicating an essentially perfect fit.
reltol	Stop if the optimizer is unable to reduce the fit criterion by a factor of at least $1 - \text{reltol}$ .
...	arguments passed to or from other methods.

object of class "nnet" or "nnet.formula". Mostly internal structure, but has components

wts	the best set of weights found.
value	value of fitting criterion plus weight decay term.
fitted.values	the fitted values for the training data.
residuals	the residuals for the training data.
convergence	1 if the maximum number of iterations was reached, otherwise 0.

## Тема 20. НМ в задачах кластеризації: карти Кохонена

Самоорганізаційні карти (SOM, Self Organizing Maps), розроблені Т. Кохоненом (Kohonen, 1982), – один з варіантів багатовимірної кластеризації зі збереженням топологічної подібності.

Мережа Кохонена має всього два шари: вхідний та вихідний, що складається з радіальних нейронів впорядкованої структури (вихідний шар називають також шаром топологічної карти). Нейрони вихідного шару розташовуються у вузлах двовимірної (об'єкт-нейрон) карти. Під час використання цього алгоритму точки, що близькі у початковому  $m$ -вимірному просторі, розміщуються поруч на карті. Кожен об'єкт повинен знаходитися в певному вузлі (в одному вузлі може бути один об'єкт, може кілька, а може взагалі не бути об'єктів). Схожі об'єкти повинні перебувати або в одному вузлі, або в сусідніх.

Отже, всі об'єкти можна подати в двомірному просторі ознак, де з кожним вузлом карти (нейроном) будуть асоціюватися локальні згущення вихідних об'єктів, що можуть бути потенційними центрами кластерів<sup>6</sup>.

Навчання шару Кохонена проводиться без вчителя, навчальна вибірка складається лише з вхідних векторів. Навчальний алгоритм підлаштовує ваги мережі так, щоб отримувати узгоджені вихідні вектори, тобто щоб подання досить близьких вхідних векторів давало одинакові виходи.

---

<sup>6</sup> Класичний алгоритм кластеризації, описаний у попередніх розділах, легко реалізується у вигляді нейронної мережі. Для оцінки близькості об'єкта до ядра кластера в алгоритмах кластеризації використовується евклідова міра близькості  $D(x^p, c^k)$ , що тим менша, чим більше об'єкт схожий на ядро кластера

$$D(x^p, c^k) = \sum_p \sum_i (x_i^p - c_i^k)^2 \rightarrow \min$$

де  $c^k$  – множина ядер  $k$  кластерів,  $x^p$  – множина об'єктів, що підлягає кластеризації.

Якщо розкріти скобки у цьому вираженні, бачимо, що перша та остання складові не впливають на якість розбиття, тому співвідношення можна переписати у такому вигляді

$$D(x^p, c^k) = \sum_p \sum_i x_i^p \times c_i^k \rightarrow \max$$

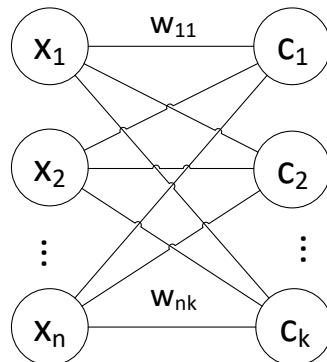
Бачимо, що  $D(x^p, c^k)$  – аналог зваженої суми  $\sum w_{ij}x_i$ , що розраховується формальним нейроном, де  $x_i$  – параметри окремого об'єкта,  $w_{ij}$  – синаптичні ваги.

Процес навчання, отже, виділяє статистичні властивості навчальної множини та групує подібні вектори в кластери. Отриманий після навчання набір карт може використовуватися для аналізу закономірностей у сукупності даних.

Під час навчання шару Кохонена на вхід подається вхідний вектор  $x$ , і обчислюються його скалярні добутки з векторами ваг  $w$ , пов'язаними з усіма нейронами Кохонена  $c$ . Нейрон з максимальним значенням скалярного добутку оголошується «переможцем», його ваги підлаштовуються

$$w_{ij}(t+1) = w_{ij}(t) + \eta(x_i - w_{ij}).$$

У такий спосіб кожна вага, пов'язана з нейроном-переможцем, змінюється пропорційно різниці між його значенням та значенням входу, до якого він приєднаний. Напрямок зміни мінімізує різницю між вагою та відповідним входом.



В середовищі R для навчання мережі зазвичай використовуються функції *somgrid()* і *som()* з пакету *kohonen*. Із завершенням ітераційного процесу для візуалізації стає доступним комплект карт з параметрами:

- *Codes* – показує на карті розподіл співвідношення часток окремих вихідних змінних;
- *Counts* – кількість вихідних об'єктів в кожному вузлі карти;
- *Mapping* – координати вихідних об'єктів на сформованій карті;
- *Property, quality, dist.neighbours* – різними кольорами зображується сукупність властивостей кожного вузла: частки окремих вихідних змінних, середні відстані між нейронами тощо.

## Лабораторна робота 17

### Kohonen maps

#### Download the data

```
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
f <- f[-1]
```

Висновок: для побудови карт Кохонена використані дані про наявність прострочених платежів по кредиту.

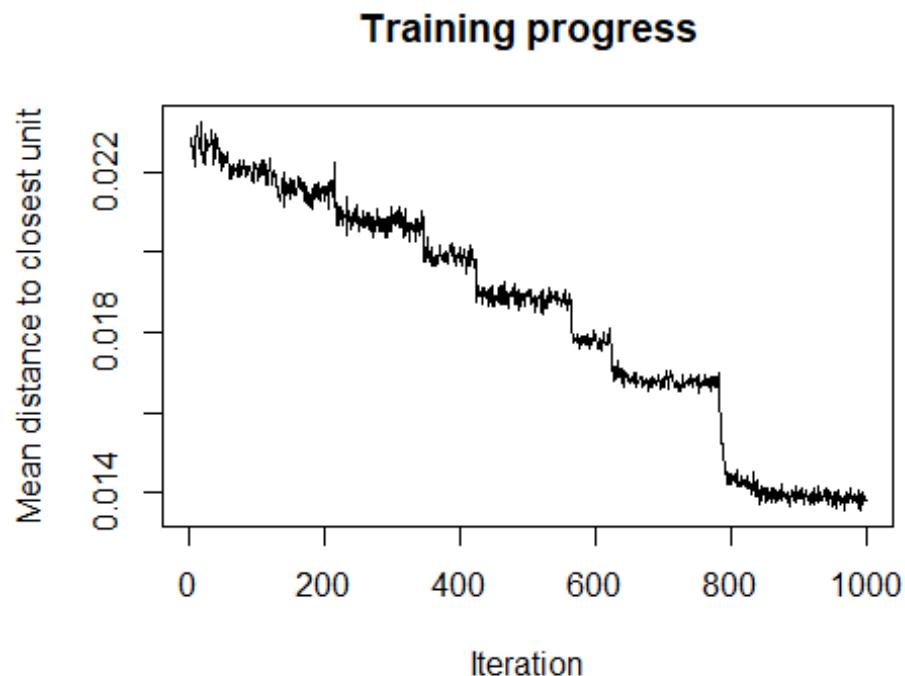
#### Features Scaling

```
f$age <- scale(f$age)
f$income <- scale(f$income)
f$children <- scale(f$children)
f$sex <- as.numeric(f$sex)
f$region <- as.numeric(f$region)
f$married <- as.numeric(f$married)
f$car <- as.numeric(f$car)
f$mortgage <- as.numeric(f$mortgage)
f$delays <- as.numeric(f$delays)
f_matrix <- as.matrix(f)
```

Висновок: модель Кохонена потребує шкалювання, виконано шкалювання кількісних змінних, якісні змінні перетворено на кількісні.

#### Fitting the NN

```
set.seed(123)
library(kohonen)
som_grid <- somgrid(xdim = 10, ydim = 6, topo = "hexagonal")
som_model <- som(f_matrix, grid = som_grid, rlen = 1000,
                  alpha = c(0.05, 0.01), keep.data = TRUE)
plot(som_model, type = "changes")
```



Висновок: графік зміни помилки нейронної мережі Кохонена свідчить про успішне навчання моделі.

## Visualization

```
#Palette
coolBlueHotRed <- function(n, alpha = 1) {
  rainbow(n, end = 4/6, alpha = alpha)[n:1]
}
par(mfrow = c(1, 2))
#Number of objects at sell
plot(som_model, type = "counts", palette.name = coolBlueHotRed)
#Distance to core
plot(som_model, type = "quality", palette.name = coolBlueHotRed)
```

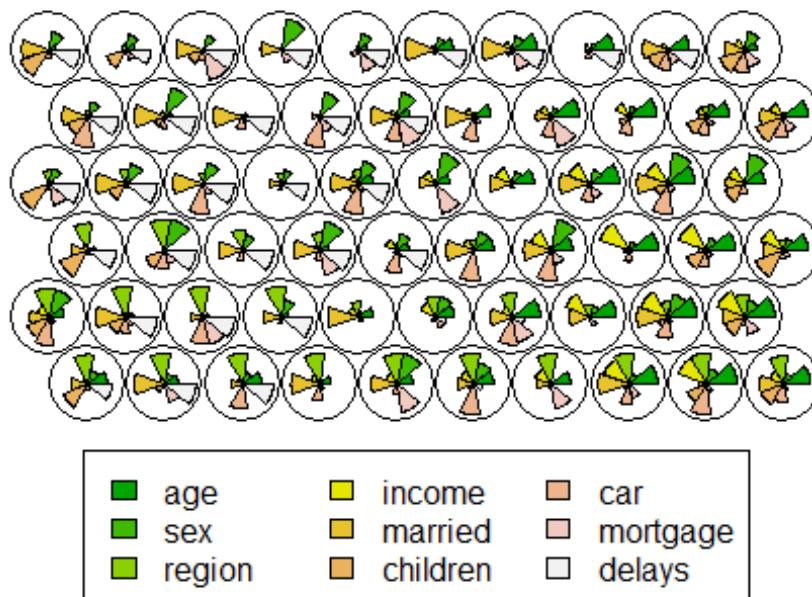


Висновок: в кожну клітинку мережі потрапило від 5 до 15 спостережень, мережа досить повна.

### Maps of the factors

```
plot(som_model, type = "codes")
```

**Codes plot**



**Висновок:** ця візуалізація дозволяє аналізувати всі фактори на одній карті.

```
par(mfrow = c(3, 3))
plot(som_model, type = "property",
      property = som_model$codes[[1]][,1],
      main = "age",
      palette.name = coolBlueHotRed)

plot(som_model, type = "property",
      property = som_model$codes[[1]][,2],
      main = "sex",
      palette.name = coolBlueHotRed)

plot(som_model, type = "property",
      property = som_model$codes[[1]][,3],
      main = "region",
      palette.name = coolBlueHotRed)

plot(som_model, type = "property",
      property = som_model$codes[[1]][,4],
      main = "income",
      palette.name = coolBlueHotRed)

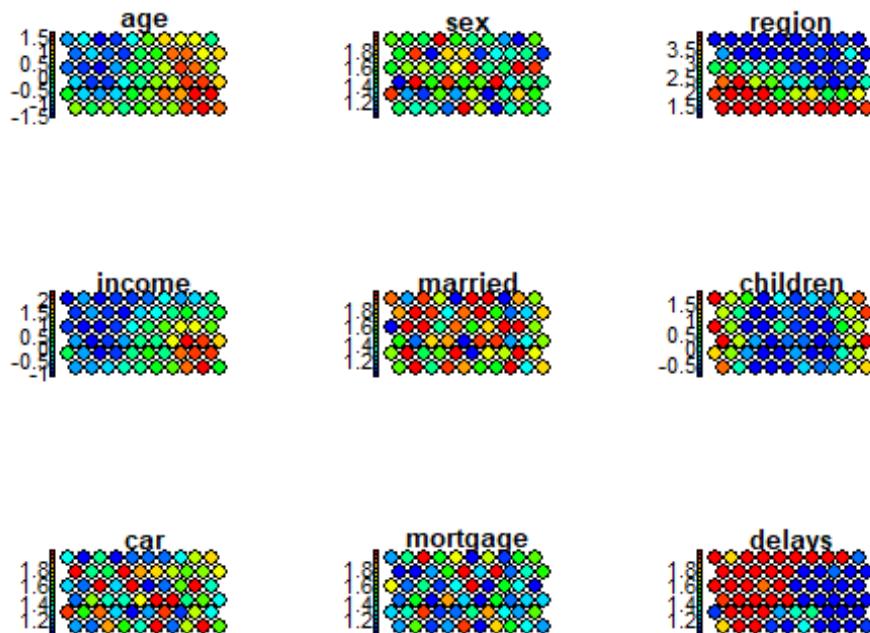
plot(som_model, type = "property",
      property = som_model$codes[[1]][,5],
      main = "married",
      palette.name = coolBlueHotRed)

plot(som_model, type = "property",
      property = som_model$codes[[1]][,6],
      main = "children",
      palette.name = coolBlueHotRed)

plot(som_model, type = "property",
      property = som_model$codes[[1]][,7],
      main = "car",
      palette.name = coolBlueHotRed)

plot(som_model, type = "property",
      property = som_model$codes[[1]][,8],
      main = "mortgage",
      palette.name = coolBlueHotRed)

plot(som_model, type = "property",
      property = som_model$codes[[1]][,9],
      main = "delays",
      palette.name = coolBlueHotRed)
```

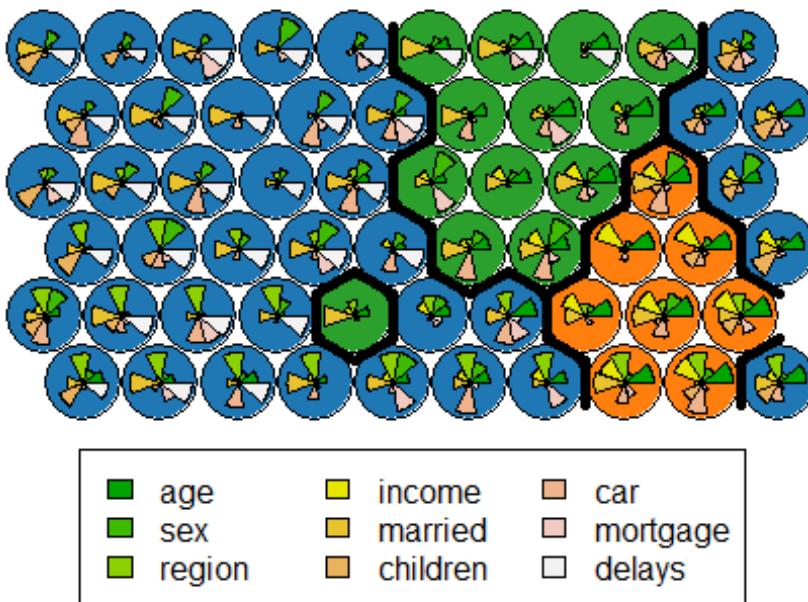


Висновок: ця візуалізація дозволяє аналізувати всі фактори на окремих картах та зробити висновок про наявність 3 кластерів.

### Clusters description

```
mydata <- as.matrix(som_model$codes[[1]])
#Use hierarchical clustering, k=3
som_cluster <- cutree(hclust(dist(mydata)), 3)
#Palette
pretty_palette <- c("#1f77b4", '#ff7f0e', '#2ca02c',
                      '#d62728', '#9467bd', '#8c564b', '#e377c2')
#Colored clusters
plot(som_model, type = "codes",
      bgcol = pretty_palette[som_cluster])
add.cluster.boundaries(som_model, som_cluster)
```

## Codes plot



**Висновок:** ця візуалізація дозволяє аналізувати параметри трьох виявленіх кластерів на одній карті.

```
aggregate(mydata, by=list(som_cluster), FUN=mean)
```

```
##   Group.1      age      sex   region     income   married   children
## 1       1 -0.4229939 1.512258 2.487115 -0.3968485 1.611838  0.2081633
## 2       2  1.4032557 1.525003 2.408360  1.9385390 1.642325  0.1488723
## 3       3  0.6215529 1.401911 1.268678  0.1860593 1.731162 -0.5715186
##           car mortgage   delays
## 1 1.490532 1.383233 1.684867
## 2 1.579548 1.303440 1.000000
## 3 1.464247 1.361836 1.330356
```

**Висновок:** на основі нейроних мереж Кохонена виявлено три кластери. Розраховано характеристики типових об'єктів кластерів.

### Довідка: som

```
som(X, ...)
xyf(X, Y, ...)
supersom(data, grid=somgrid(), rlen = 100, alpha = c(0.05, 0.01),
          radius = quantile(nhbrdist, 2/3),
          whatmap = NULL, user.weights = 1, maxNA.fraction = 0L,
          keep.data = TRUE, dist.fcts = NULL,
          mode = c("online", "batch", "pbatch"), cores = -1, init,
          normalizeDataLayers = TRUE)
```

## Arguments

x, y	numerical data matrices, or factors. No data.frame objects are allowed – convert them to matrices first.
data	list of data matrices (numerical) of factors. If a vector is entered, it will be converted to a one-column matrix. No data.frame objects are allowed.
grid	a grid for the codebook vectors: see somgrid.
rlen	the number of times the complete data set will be presented to the network.
alpha	learning rate, a vector of two numbers indicating the amount of change. Default is to decline linearly from 0,05 to 0,01 over rlen updates. Not used for the batch algorithm.
radius	the radius of the neighbourhood, either given as a single number or a vector (start, stop). If it is given as a single number the radius will change linearly from radius to zero; as soon as the neighbourhood gets smaller than one only the winning unit will be updated. Note that the default before version 3.0 was to run from radius to -radius. If nothing is supplied, the default is to start with a value that covers 2/3 of all unit-to-unit distances.
whatmap	What data layers to use. If unspecified all layers are used.
user.weights	the weights given to individual layers. This can be a single number (all layers have the same weight, the default), a vector of the same length as the whatmap argument, or a vector of the same length as the data argument. In xyf maps, this argument provides the same functionality as the now-deprecated xweight argument that was used prior to version 3.0.
maxNA.fraction	the maximal fraction of values that may be NA to prevent the row to be removed.
keep.data	if TRUE, return original data and mapping information. If FALSE, only return the trained map (in essence the codebook vectors).
dist.fcts	vector of distance functions to be used for the individual data layers, of the same length as the data argument, or the same length of the whatmap argument. If the length of this vector is one, the same distance will be used for all layers. Admissible values currently are "sumofsquares", "euclidean", "manhattan", and "tanimoto". Default is to use "sumofsquares" for continuous data, and "tanimoto" for factors.
mode	type of learning algorithm.
cores	number of cores to use in the "pbatch" learning mode. The default, -1, corresponds to using all available cores.
init	list of matrices, initial values for the codebook vectors. The list should have the same length as the data list, and corresponding

numbers of variables (columns). Each list element should have a number of rows corresponding to the number of units in the map.

`normalizeDataLayers` boolean, indicating whether `distance.weights` should be calculated (see details section). If `normalizeDataLayers == FALSE` the user weights are applied to the data immediately.

`...` Further arguments for the supersom function presented to the som or xyf wrappers.

An object of class "kohonen" with components

<code>data</code>	data matrix, only returned if <code>keep.data == TRUE</code> .
<code>unit.classif</code>	winning units for all data objects, only returned if <code>keep.data == TRUE</code> .
<code>distancess</code>	distances of objects to their corresponding winning unit, only returned if <code>keep.data == TRUE</code> .
<code>grid</code>	the grid, an object of class <code>somgrid</code> .
<code>codes</code>	a list of matrices containing codebook vectors.
<code>changes</code>	matrix of mean average deviations from code vectors; every map corresponds with one column.
<code>alpha, radius, user.weights, whatmap, maxNA.fraction</code>	input arguments presented to the function.
<code>distance.weights</code>	if <code>normalizeDataLayers</code> weights to equalize the influence of the individual data layers, else a vector of ones.
<code>dist.fcts</code>	distance functions corresponding to all layers of the data, not just the ones indicated by the <code>whatmap</code> argument.

### **Питання для самоперевірки**

1. Модель штучного нейрона.
2. Види функцій активації.
3. Властивості й обмеженість моделі персептрона.
4. Вибір архітектури НМ. Архітектура НМ прямої передачі сигналу, що звужується та розширюється.
5. Навчання з учителем і без нього.
6. Ухвалення рішення про закінчення навчання на основі помилок навчання й тестування.
7. Особливості використання НМ в задачах апроксимації.
8. Особливості використання НМ в задачах прогнозування.
9. Особливості використання НМ в задачах класифікації.
10. Особливості використання НМ в задачах кластеризації.

### **Самостійна робота 7**

A. Зібрати дані, що підлягають прогнозуванню, побудувати НМ, зробити висновки про якість моделі.

B. Зібрати дані, що підлягають класифікації, побудувати НМ-класифікатор, зробити висновки про якість моделі.

C. Зібрати дані про динаміку будь-якого процесу (наприклад валютного курсу, пошукового запиту). Побудувати НМ для прогнозування обраного індикатора, зробити висновки про якість моделі.

D. Зібрати дані, що підлягають кластеризації, побудувати відповідну НМ, зробити висновки про якість моделі.

## РОЗДІЛ 8. ГЛИБОКЕ НАВЧАННЯ

Глибоке навчання – підмножина методів машинного навчання, в яких застосовуються багатошарові (що складаються з десятків і сотень шарів) нейронні мережі складної архітектури (не повнозв'язані, зі зворотними зв'язками). Своїми успіхами глибоке навчання зобов'язане експоненціальному зростанню як потужностей (зокрема дешевих графічних процесорів), що використовуються для їх побудови, так і обсягів даних (наприклад, ImageNet для обробки зображень), що можуть використовуватися для навчання подібних архітектур.

Для роботи з глибокими НМ будемо використовувати Keras – відкриту нейромережеву бібліотеку, написану на мові Python (зокрема R-інтерфейсів до TensorFlow і Keras). Вона була створена в рамках проекту ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).

Keras надає високорівневий, інтуїтивний набір абстракцій, що спрощує формування нейронних мереж незалежно від використовуваної в якості обчислювального бекенду бібліотеки наукових обчислень. Ця бібліотека містить численні реалізації широко застосовуваних блоків нейронних мереж, таких як шари, цільові та передавальні функції, оптимізатори, та купу інструментів для спрощення роботи з зображеннями та текстом. Її код розміщений на GitHub, а форуми підтримки містять сторінку питань GitHub, канали Gitter і Slack.

Для обчислень з тензорами – «будівельний блок» нейронних мереж – Keras використовує бібліотеки Theano або TensorFlow, що створені для роботи з багатовимірними масивами (тензорами). На їх основі можливо створювати моделі як з послідовною, так і з функціональною архітектурою.

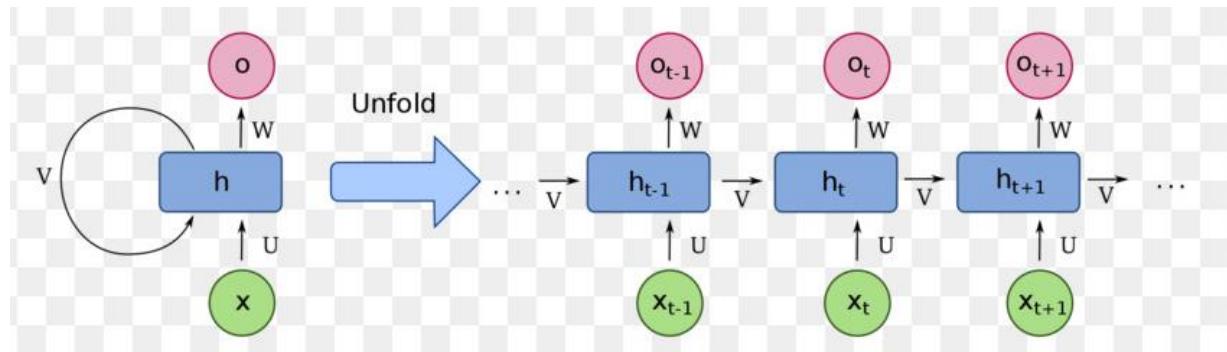
## Тема 21. Рекурентні нейронні мережі

Рекурентні нейронні мережі (РНМ, Recurrent neural network, RNN) – клас НМ, в яких використовується послідовна природа вхідних даних. Такими даними можуть бути часові ряди, текст, мова або будь-який інший об'єкт, в якому появляється наступного елемента послідовності залежить від попередніх елементів.

Основним припущенням багатошарового персептрона є незалежність входів. Однак для послідовних даних це припущення порушується. Для подолання цієї проблеми в РНМ додається блок пам'яті, в якому зберігається попередня інформація

$$o_t = h(o_{t-1}, x_t),$$

де  $o_t$  – значення пам'яті на поточному кроці,  $x_t$  – вхідне значення в поточний момент часу. Це співвідношення рекурентне,  $o_{t-1}$  можна виразити через  $o_{t-2}$ , тобто відновити інформацію як завгодно довгої послідовності.



Якщо параметри персептрона зберігаються в матриці ваг, то параметри РНМ задаються трьома матрицями  $U$ ,  $V$  і  $W$ , що відповідають входу, виходу і блоку пам'яті. Ці матриці розділяються між усіма кроками, оскільки на кожному кроці до різних даних застосовуються одні і ті ж операції

$$h_t = \tanh(Vh_{t-1} + Ux_t)$$

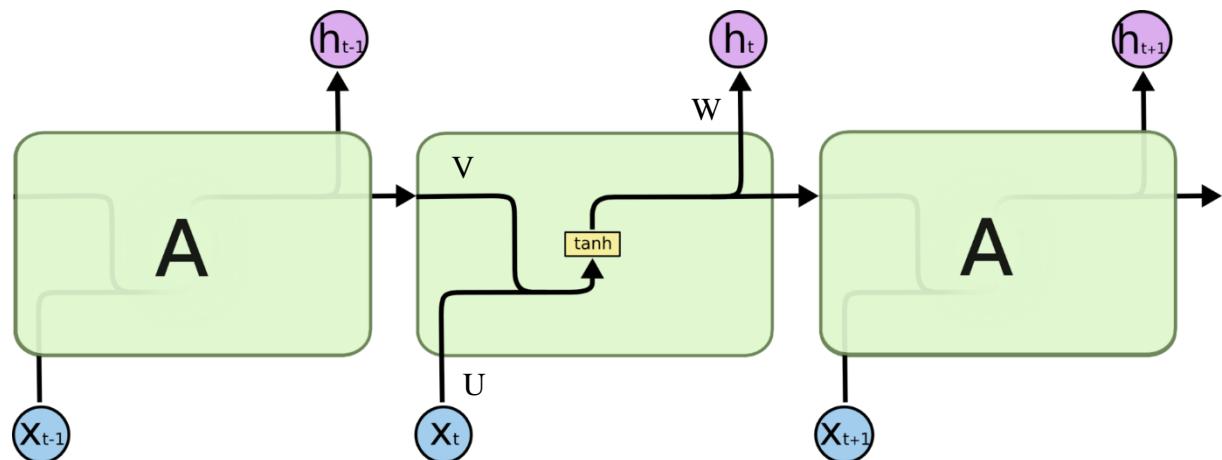
$$o_t = \text{softmax}(Wh_{t-1})$$

Завдяки використанню одних і тих же ваг на всіх етапах вдається істотно знизити кількість параметрів РНМ.

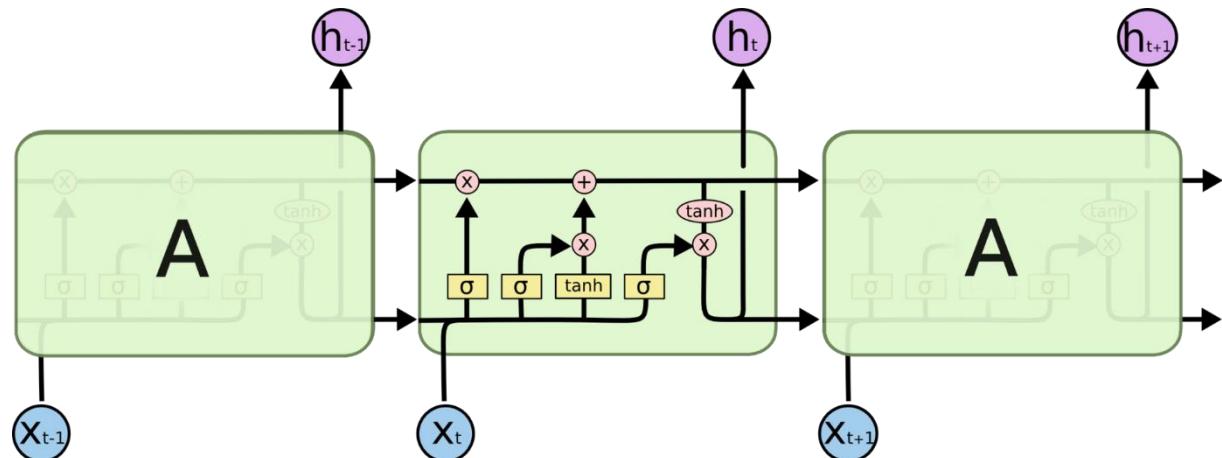
Недолік РНМ полягає в такому: якщо враховувати кожен крок часу, то для кожного кроку часу необхідно створювати свій шар нейронів, що викликає серйозні труднощі в обчисленні. Крім того, багатошарові реалізації виявляються обчислювально нестійкими, тому що в них зазвичай зникають або зашкалюють ваги. Проте, якщо обмежити розрахунок фіксованим часовим вікном, то отримані моделі не відображають довгострокових трендів.

Мережа з довготривалою і короткочасною пам'яттю (Long short term memory, LSTM) дозволяє вирішити цю проблему.

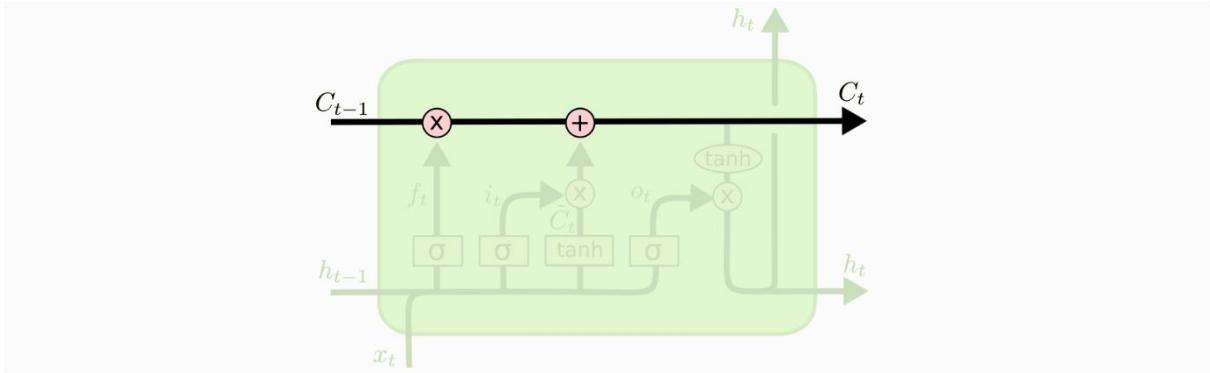
У класичній РНМ рекурентність реалізована як комбінація блоку пам'яті про попередній крок і поточних входних даних, загорнута в функцію активації



Для того, щоб вказувати, що слід пам'ятати, а що забути, LSTM містить не один, а чотири шари

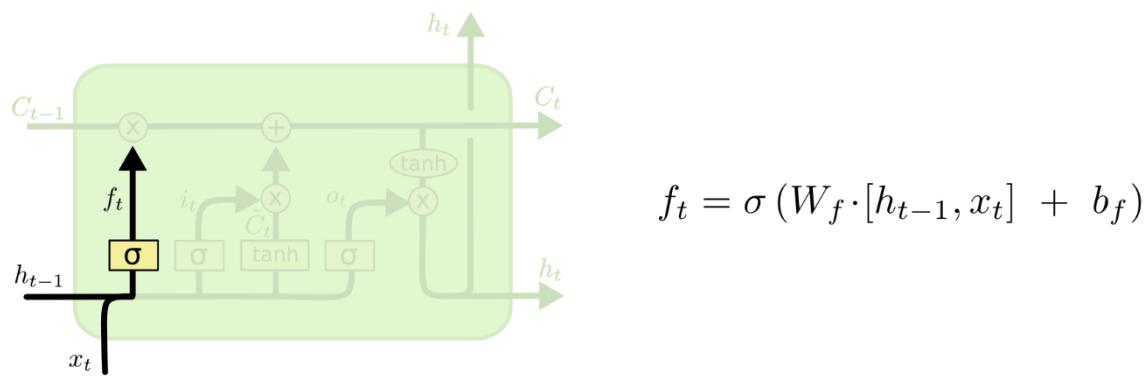


На горизонтальній лінії, що проходить по верхній частині схеми відображається стан комірки (cell state) – це внутрішня пам'ять блоку.

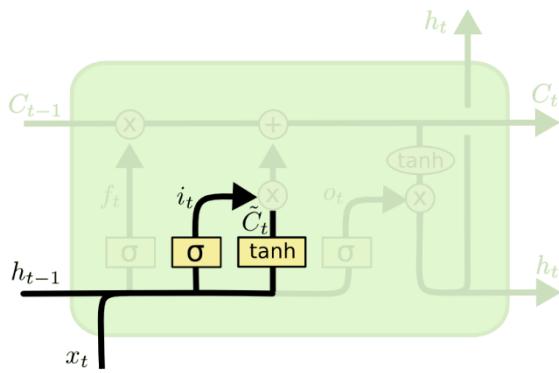


Видалення інформації регулюється структурами, так званими фільтрами (gates). Фільтри дозволяють пропускати інформацію на підставі деяких умов. Вони складаються з сигмоїdalного шару та операції поточкового множення. Сигмоїdalний шар повертає числа від нуля до одиниці, що позначають, яку частку кожного блоку інформації варто пропустити далі по мережі. Нуль у цьому разі означає «не пропускати нічого», одиниця – «пропустити все». У LSTM три такі фільтри, що дозволяють захищати і контролювати стан комірки.

Рішення про те, яку інформацію можна вилучити зі стану комірки, ухвалює сигмоїdalний шар, що називається «шаром фільтра забування» (forget gate layer).



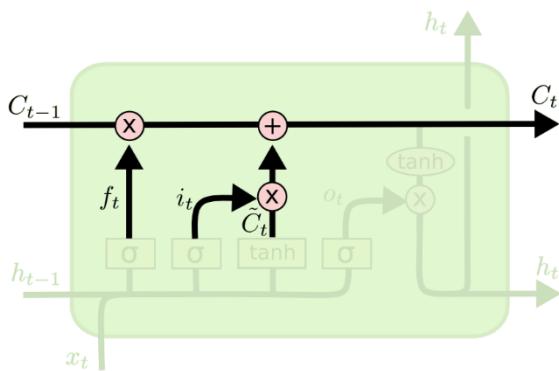
Для рішення про те, яка нова інформація буде зберігатися в стані комірки, спочатку сигмоїdalний шар під назвою «шар вхідного фільтра» (input layer gate) визначає, які значення слід оновити. Потім tanh-шар буде вектор нових значень-кандидатів, що можна додати до стану комірки.



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

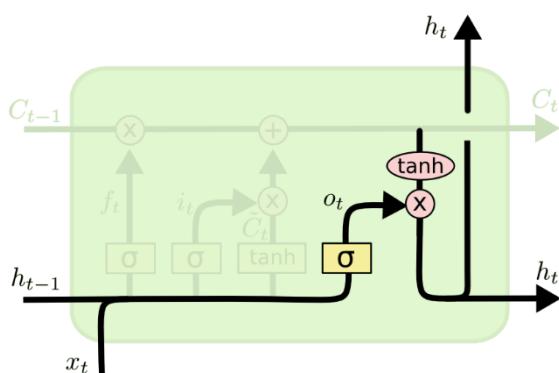
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Щоб замінити старий стан комірки на новий виконується така операція



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Вихідні дані будуть організовані на новому стані комірки з урахуванням фільтрів – спочатку застосовується сигмоїdalний шар, що вирішує, яку інформацію зі стану комірки виводити; потім значення стану комірки проходять через tanh-шар, щоб отримати на виході значення з діапазону від -1 до 1, і перемножуються з вихідними значеннями сигмоїdalного шару, що дозволяє виводити тільки необхідну інформацію.



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

## Лабораторна робота 18

# DEEP LEARNING, LSTM

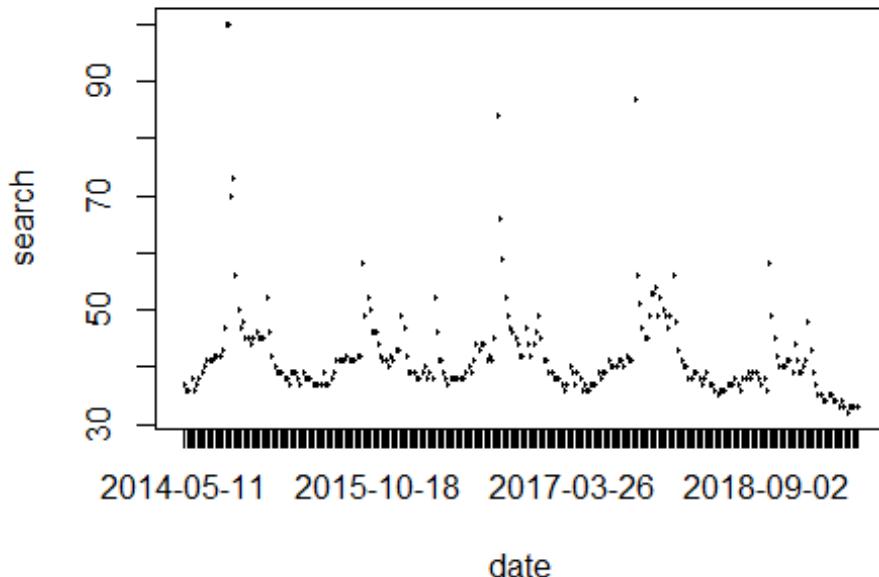
### Downloading libraries

```
#devtools::install_github("rstudio/keras")
# then install Tensorflow backend as follows:
library(keras)
#install_keras()
library(tensorflow)
```

### Downloading data

```
#Set Working Directory
setwd('D:/ML/')
#OR Choose your Directory in 'Files' and click on 'More' -> 'Set as
Working Directory'

#Download file to the table.
f <- read.csv('iPhone.csv', header = TRUE, encoding = 'UNICOD')
plot(f)
```



Висновок: завантажено часовий ряд.

## Transforming data

```
#to stationarity
df = diff(f$search, differences = 1)
df <- as.data.frame(df)
#train set size
N = nrow(df)
n = round(N *0.7, digits = 0)
#scaling training set
ymax <- max(df$df[1:n])
ymin <- min(df$df[1:n])
df$y <- 0
df$y[1:n] <- (df$df[1:n] - ymin ) / (ymax - ymin)
#scaling testing set
ymax <- max(df$df[(n+1):N])
ymin <- min(df$df[(n+1):N])
df$y[(n+1):N] <- (df$df[(n+1):N] - ymin ) / (ymax - ymin)
df$x <- dplyr::lag(df$y)
head (df)

##   df          y          x
## 1 -1 0.3571429      NA
## 2  0 0.3690476 0.3571429
## 3  2 0.3928571 0.3690476
## 4 -2 0.3452381 0.3928571
## 5  1 0.3809524 0.3452381
## 6  1 0.3809524 0.3809524
```

Висновок: ряд зведений до стаціонарного, виконано шкалювання.

## Splitting the dataset into the TRAIN set and TEST set

```
train = df[2:n, ]
test = df[(n+1):N, ]
x_train <- train$x
y_train <- train$y
x_test <- test$x
y_test <- test$y
```

Висновок: датасет розподілений на навчальну та тестову вибірки послідовно.

## Defining the model

```
#We set the argument stateful = TRUE so that the internal states obtained  
after processing a batch of samples are reused as initial states for the  
samples of the next batch. Since the network is stateful, we have to  
provide the input batch in 3-dimensional array of the form [samples,  
timesteps, features] from the current [samples, features], where:  
#Samples: Number of observations in each batch, also known as the batch  
size.  
#Timesteps: Separate time steps for a given observations. In this example  
the timesteps = 1  
#Features: For a univariate case, like in this example, the features = 1  
#The batch size must be a common factor of sizes of both the training and  
testing samples. 1 is always a sure bet.  
=====  
=====  
# Reshape the input to 3-dim  
dim(x_train) <- c(length(x_train), 1, 1)  
  
# specify required arguments  
X_shape2 = dim(x_train)[2]  
X_shape3 = dim(x_train)[3]  
batch_size = 1 # must be a common factor of both the train and test  
samples  
units = 1      # can adjust this, in model tuning phase  
=====  
model <- keras_model_sequential()  
model%>%  
  layer_lstm(units, batch_input_shape = c(batch_size, X_shape2, X_shape3),  
stateful = TRUE)%>%  
  layer_dense(units = 1)
```

## Compiling the model

```
#Here we have specified the mean_squared_error as the loss function,  
Adaptive Moment Estimation (ADAM) as the optimization algorithm and  
Learning rate and Learning rate decay over each update. Finally, we have  
used the accuracy as the metric to assess the model performance.  
model %>% compile(  
  loss = 'mean_squared_error',  
  optimizer = optimizer_adam( lr = 0.02, decay = 1e-6 ),  
  metrics = c('accuracy'))
```

## Model summary

```
summary(model)
```

```
## _____
## Layer (type)          Output Shape      Param #
## =====
## lstm (LSTM)           (1, 1)           12
## _____
## dense (Dense)         (1, 1)           2
## =====
## Total params: 14
## Trainable params: 14
## Non-trainable params: 0
## _____
```

## Fitting the model

*#We set the argument shuffle = FALSE to avoid shuffling the training set and maintain the dependencies between  $x_i$  and  $x_{i+1}$ . LSTM also requires resetting of the network state after each epoch. To achieve this we run a loop over epochs where within each epoch we fit the model and reset the states via the argument reset\_states().*

```
Epochs = 50
for(i in 1:Epochs ){
  model %>% fit(x_train, y_train, epochs = 1, batch_size = batch_size,
  verbose = 1, shuffle = FALSE)
  model %>% reset_states()
}
```

## Making predictions

```
L = length(y_test)
predictions = numeric(L)

for(i in 1:L){
  X = x_test[i]
  dim(X) = c(1,1,1)
  yhat = model %>% predict(X, batch_size = batch_size)
  predictions[i] <- yhat
}

mse_lstm<-sum((y_test-predictions)^2)/L
mse_lstm

## [1] 0.01034692
```

Висновок: значення середньоквадратичної помилки на тестовій вибірці – 0,01.

## Inverting

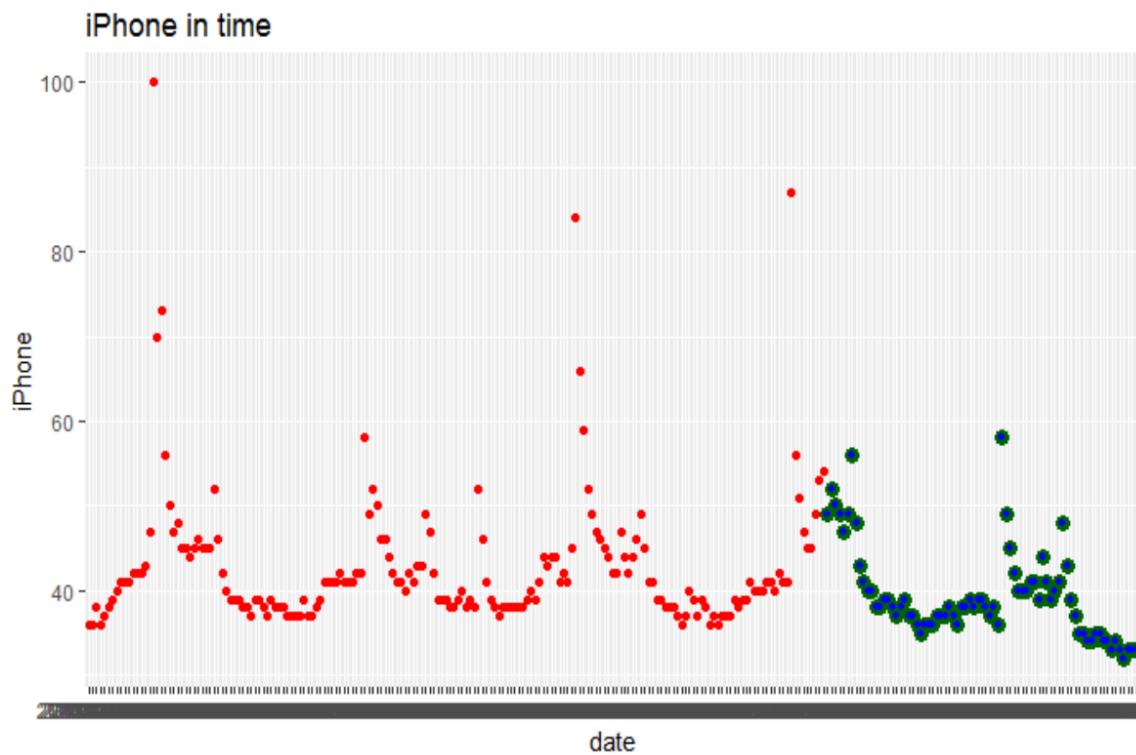
```
# invert scaling
ys = predictions * (ymax - ymin) + ymin
# invert differencing
fd = numeric(N-n-1)
fd[1] <- f$search[n+1] + ys[1]
for(i in 2:(N-n-1)){
```

```
    fd[i] = fd[i-1] + ys[i]
}
```

Висновок: виконано зворотне перетворення даних.

## Visualising

```
library(ggplot2)
ggplot() +
  geom_point(aes(f$date[2:(n+1)], f$search[2:(n+1)]), colour = 'red') +
  geom_point(aes(f$date[(n+2):N], f$search[(n+2):N]), colour = 'dark
green', size = 3) +
  geom_point(aes(f$date[(n+2):N], fd), colour = 'blue', size = 1) +
  ggtitle('iPhone in time') +
  xlab('date') +
  ylab('iPhone')
```



Висновок: на графіку червоним позначені точки навчальної вибірки, зеленим – точки тестової вибірки, синім – модельні значення.

## Довідка: keras\_model\_sequential

```
keras_model_sequential(layers = NULL, name = NULL)
```

### Arguments

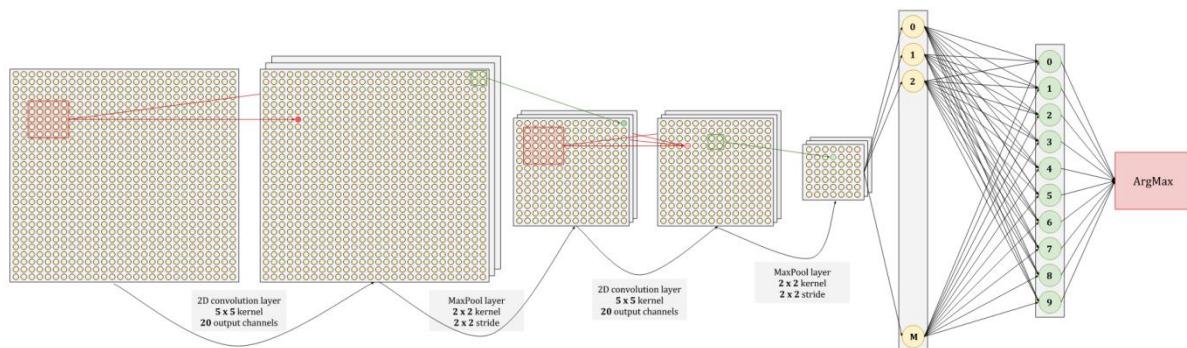
layers List of layers to add to the model

name Name of model

## Тема 22. Згорткові нейронні мережі

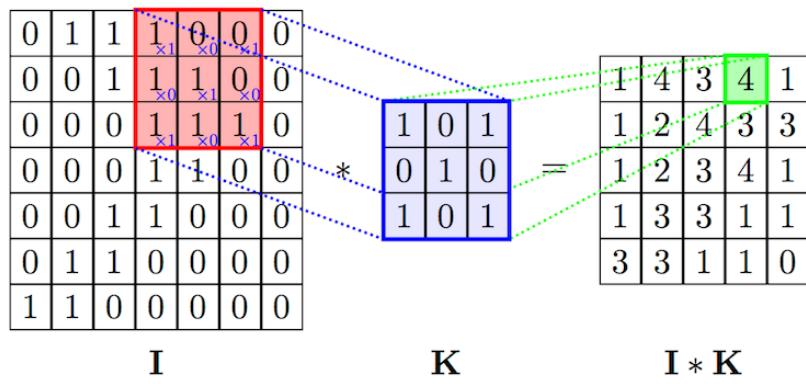
Згорткові мережі – клас НМ, в яких використовується просторова інформація, закладена в даних. Вони інспіровані фізіологічними експериментами із зоровою корою, мають складну багатошарову архітектуру і використовуються для навчання моделей з підвищеним рівнем абстракції (зокрема, для вирішення завдань розпізнавання образів за аналогією із зором).

Глибока згорткова нейронна мережа (ГЗНМ) складається з шарів двох типів (в розпізнавальній частині архітектури) – згортальних і пулінгових, комбінація з яких повторюється необхідну кількість разів; а також одного або декількох повнозв'язаних шарів, що використовувалися на останніх етапах для вирішення задачі класифікації.

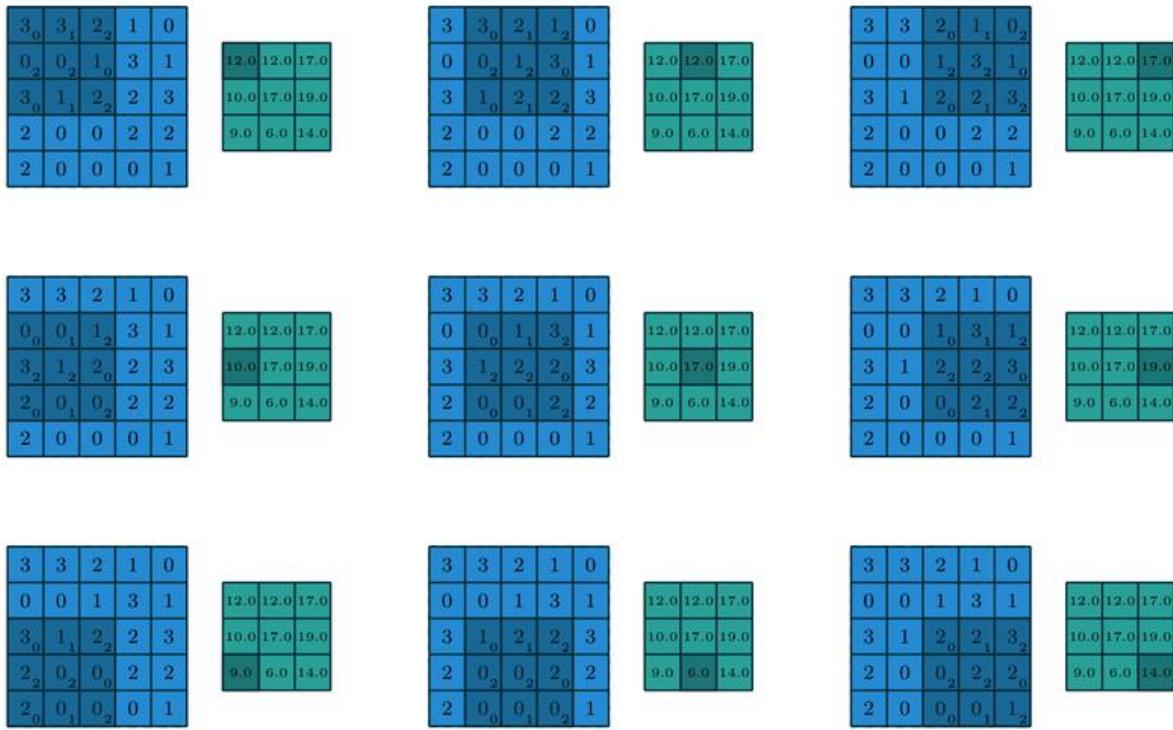


Концепція ГЗНМ побудована на поняттях локального рецептивного поля, розподілених ваг і пулінгу. Розглянемо їх докладніше.

Для збереження просторової інформації, зображення зручно подавати як матрицю пікселів. Для кодування локальної структури модматриця сусідніх входних нейронів з'єднується з одним прихованим нейроном наступного шару, що являє собою локальне *рецептивне поле*. Ця операція називається згорткою.

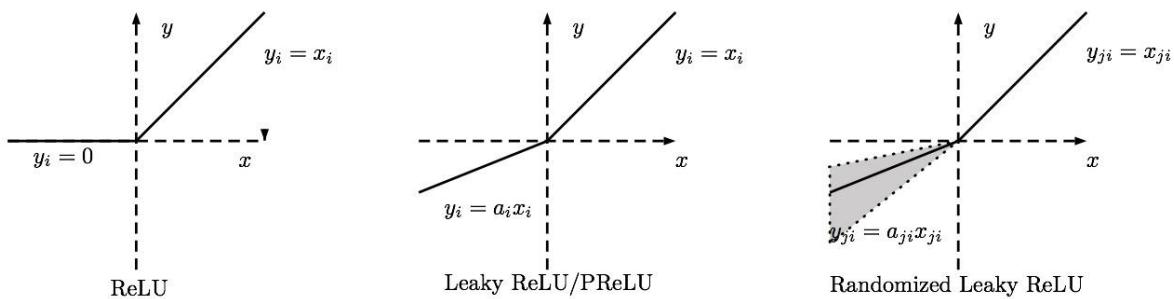


Для того, щоб знаходити одну і ту саму ознаку (наприклад, горизонтальну межу) незалежно від того, в якому місці зображення вона перебуває, використовується загальна сукупність ваг і зміщень для всіх нейронів у прихованому шарі. У цьому разі кожен нейрон вчиться розпізнавати множину позиційно-незалежних ознак у зображенні.



У такий спосіб шар згортки містить фільтр для кожного каналу, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати матричного добутку для кожного фрагмента). Вагові коефіцієнти ядра згортки (невеликої матриці) заздалегідь невідомі і встановлюються в процесі навчання.

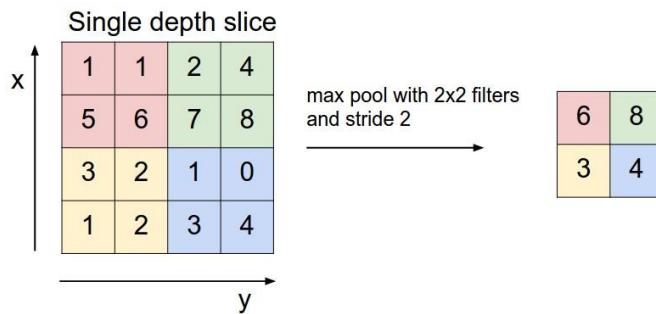
Скалярний результат кожної згортки потрапляє на функцію активації, що являє собою певну нелінійну функцію. Шар активації зазвичай логічно пов'язують із шаром згортки (вважають, що функція активації вбудована у шар згортки). Функція нелінійності може бути будь-якою за вибором дослідника, традиційно для цього використовували функції типу гіперболічного тангенса. Однак останнім часом все більшої популярності набуває функція ReLU (rectified linear unit), що дозволила суттєво прискорити процес навчання і водночас спростити обчислення.



Шар пулінгу (інакше підвибірки, субдискретизація) являє собою нелінійне ущільнення карти ознак, водночас група пікселів ущільнюється до одного пікселя, проходячи нелінійне перетворення.

Пулінговий шар покликаний знижувати розмірність зображення. Початкове зображення ділиться на блоки розміром  $m \times n$ , і для кожного блоку обчислюється деяка функція. Найчастіше використовується функція максимуму (max pooling) або середнього (average pooling). Навчаних параметрів у цього шару немає. Основні цілі пулінгового шару:

- зменшення зображення;
- збільшення інваріантності виходу мережі щодо малого зсуву входу;
- прискорення обчислень.



Операція пулінг може бути інтерпретована так: якщо на операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібно, і воно ущільнюється до менш докладного. До того ж фільтрація вже непотрібних деталей допомагає мережі не перенавчатися. Шар пулінга зазвичай вставляється після шару згортки перед шаром наступної згортки.

Після кількох проходжень згортки зображення і ущільнення за допомогою пулінга система перебудовується від конкретної сітки пікселів з високою роздільною здатністю до більш абстрактних карт ознак зазвичай на кожному наступному шарі збільшується кількість каналів і зменшується розмірність зображення в кожному каналі. Зрештою залишається велика сукупність каналів, що зберігають невелику кількість даних, що інтерпретуються як абстрактні поняття, виявлені з вихідного зображення.

Ці дані об'єднуються і передаються на звичайну повнозв'язну нейронну мережу, що теж може складатися з декількох шарів. При цьому повнозв'язні шари вже втрачають просторову структуру пікселів і мають порівняно невелику розмірність.

Отже, архітектура згорткових нейронних мереж складається зі згорткових шарів, що чергуються (convolution layers) і субдискретизуючих шарів (subsampling layers або pooling layers), а також повнозв'язного шару- класифікатора в кінці. Структура мережі – односпрямована (без зворотних зв'язків), принципово багатошарова. Для навчання використовуються стандартні методи, найчастіше – метод зворотного поширення помилки.

## Лабораторна робота 19

### DEEP LEARNING, CNN

#### Downloading libraries

```
library(keras)
```

##### Data Preparation

```
batch_size <- 128
num_classes <- 10
epochs <- 12

# Input image dimensions
img_rows <- 28
img_cols <- 28

# The data, shuffled and split between train and test sets
mnist <- dataset_mnist()
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y

# Redefine dimension of train/test inputs
x_train <- array_reshape(x_train, c(nrow(x_train), img_rows, img_cols, 1))
x_test <- array_reshape(x_test, c(nrow(x_test), img_rows, img_cols, 1))
input_shape <- c(img_rows, img_cols, 1)

# Transform RGB values into [0,1] range
x_train <- x_train / 255
x_test <- x_test / 255

cat('x_train_shape:', dim(x_train), '\n')
## x_train_shape: 60000 28 28 1

cat(nrow(x_train), 'train samples\n')
## 60000 train samples

cat(nrow(x_test), 'test samples\n')
## 10000 test samples

# Convert class vectors to binary class matrices
y_train <- to_categorical(y_train, num_classes)
y_test <- to_categorical(y_test, num_classes)
```

## Defining model

```
model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 32, kernel_size = c(3,3), activation = 'relu',
    input_shape = input_shape) %>%
  layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu')
%>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = num_classes, activation = 'softmax')
```

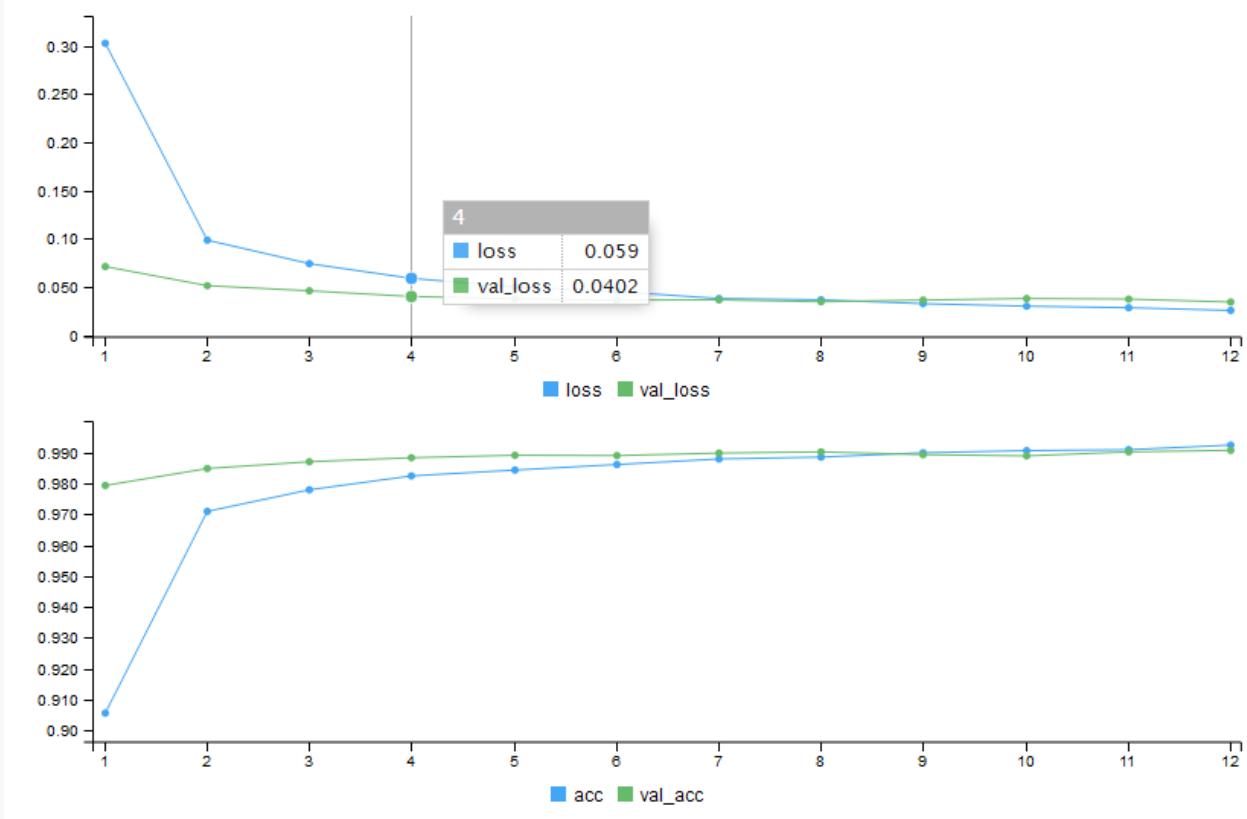
## Compiling model

```
model %>% compile(
  loss = loss_categorical_crossentropy,
  optimizer = optimizer_adadelta(),
  metrics = c('accuracy'))
)
```

## Training model

```
model %>% fit(
  x_train, y_train,
  batch_size = batch_size,
  epochs = epochs,
  validation_split = 0.2
)

scores <- model %>% evaluate(
  x_test, y_test, verbose = 0
)
```



## Output metrics

```
cat('Test loss:', scores[[1]], '\n')
## Test loss: 0.02930561
cat('Test accuracy:', scores[[2]], '\n')
## Test accuracy: 0.9915
```

## Питання для самоперевірки

1. Особливості моделей глибокого навчання.
2. Бібліотека Keras.
3. Рекурентні нейронні мережі, переваги і недоліки.
4. Мережа з довготривалою і короткочасною пам'яттю, переваги і недоліки.
5. Поняття згортки.
6. Поняття пулінгу.

**Самостійна робота 8**

- A. Зібрати дані про динаміку будь-якого процесу (наприклад валютного курсу, пошукового запиту). Побудувати LSTM для прогнозування обраного індикатора, зробити висновки про якість моделі.
- B. Побудувати згорткову мережу розпізнавання рукописних цифр, зробити висновки про якість моделі.

## ДОДАТКИ

### Додаток 1. Функції в R

#### Допомога

`help (topic), ? topic` – довідка про *topic*

`help (package =)` – довідка за заданим пакетом

`help.start ()` – запустити допомогу в браузері

`example (topic)` – приклади використання *topic*

#### Поточне оточення

`ls ()` – список всіх об'єктів

`rm (x)` – видалити об'єкт

`dir ()` – показати всі файли в поточному каталогі

`getwd ()` – отримати поточну директорію

`setwd (dir)` – змінити поточну директорію на *dir*

#### Спільна робота з об'єктами

`str (object)` – внутрішня структура об'єкта *object*

`summary (object)` – загальна інформація про об'єкт *object*

`dput (x)` – отримати уявлення об'єкта в R-синтаксисі

`head (x)` – подивитися початкові рядки об'єкта

`tail (x)` – подивитися останні рядки об'єкта

#### Введення і виведення

`library (package)` – підключити пакет *package*

`save (file, ...)` – зберігає зазначені об'єкти в *XDR*-форматі, що не залежить від платформи

`load ()` – завантажує дані, збережені раніше за допомогою команди `save()`

`read.table` – читає таблицю даних і створює за ними *data.frame*

`write.table` – друкує об'єкт, конвертуючи його в *data.frame*

`read.csv` – читує *csv*-файл

`read.delim` – зчитування даних, розділених знаками табуляції

`save.image` – зберігає всі об'єкти в файл

`cat (... , file = , sep = )` – зберігає аргументи, конкатенуючи їх

через *sep*

`sink (file)` – виводить результати виконання інших команд в файл в режимі реального часу до моменту виклику цієї ж команди без аргументів

## Створення об'єктів

`from: to` – генерує послідовність чисел від *from* до *to* з кроком 1, наприклад *1 : 3*

`c (...)` – об'єднує аргументи на вектор, наприклад *c (1, 2, 3)*

`seq (from, to, by = )` – генерує послідовність чисел від *from* до *to* з кроком *by*

`seq (from, to, len = )` – генерує послідовність чисел від *from* до *to* довжини *len*

`rep (x, times)` – повторює *x* рівно *times* раз

`list (... )` – створює список об'єктів

`data.frame (... )` – створює фрейм даних

`array (data, dims)` – створює з *data* багатовимірний масив розмірності *dim*

`matrix (data, nrow = , ncol = , byrow = )` – створює з *data* матрицю *nrow* на *ncol*, порядок заповнення визначається *byrow*

`factor (x, levels = )` – створює з *x* фактор з рівнями *levels*

`gl (n, k, length = n * k, labels = 1: n)` – створює фактор з *n* рівнів, кожен з яких повторюється *k* разів довжини *length* з іменами *labels*

`rbind (... )` – об'єднує аргументи за рядками

`cbind ( . . . )` – об’єднує аргументи за стовпцями

## Індексування

### Вектори

`x [n]` –  $n$ -ий елемент

`x [-n]` – всі елементи, крім  $n$ -го

`x [1:n]` – перші  $n$  елементів

`x [- (1:n)]` – всі елементи, крім перших  $n$

`x [c (1,4,2)]` – елементи з заданими індексами

`x ["name"]` – елемент із заданим ім’ям

`x [x>3]` – всі елементи, більш 3

`x [x>3 & x<5]` – всі елементи між 3 і 5

`x [x%in% c ("a", "and", "the")]` – елементи заданої множини

### Списки

`x [n]` – список, що складається з елемента  $n$

`x [[n]]` –  $n$ -ий елемент списку

`x [[ "name"]]` – елемент списку з ім’ям *name*

`x $ name` – елемент списку з ім’ям *name*

### Матриці

`x [i, j]` – елемент на перетині  $i$ -го рядка та  $j$ -го стовпця

`x [i, ]` –  $i$ -й рядок

`x [ , j]` –  $j$ -ий стовпець

`x [ , c (1,3)]` – задана підмножина стовпців

`x [ "name", ]` – рядок з ім’ям *name*

### Фрейми

`x [ [ "name"]]` – стовпець з ім’ям *name*

`x $ name` – стовпець з ім’ям *name*

### Робота зі змінними

`as.array (x)`, `as.data.frame (x)`, `as.numeric (x)`,  
`as.logical (x)`, `as.complex (x)`, `as.character (x)` –

перетворення змінної до заданого типу

`is .na (x)`, `is.null (x)`, `is.array (x)`, `is.data.frame (x)`,  
`is.numeric (x)`, `is.complex (x)`, `is.character (x)` – перевірка  
на те, що цей об'єкт має вказаний тип

`length (x)` – кількість елементів в  $x$

`dim (x)` – розмірність об'єкта  $x$

`dimnames (x)` – імена розмірностей об'єкта  $x$

`names (x)` – імена об'єкта  $x$

`nrow (x)` – кількість рядків  $x$

`ncol (x)` – кількість стовпців  $x$

`class (x)` – клас об'єкта  $x$

`unclass (x)` – видаляє атрибут класу у об'єкта  $x$

`attr (x, which)` – атрибут  $which$  об'єкта  $x$

`attributes (obj)` – список атрибутів об'єкта  $obj$

### Маніпуляція даними

`which.max (x)` – індекс елемента з максимальним значенням

`which.min (x)` – індекс елемента з мінімальним значенням

`rev (x)` – реверсує порядок елементів

`sort (x)` – сортує елементи об'єкта за зростанням

`cut (x, breaks)` – ділить вектор на рівні інтервали

`match (x, y)` – шукає елементи  $x$ , які є в  $y$

`which (x == a)` – повертає порядкові елементи  $x$ , які дорівнюють  $a$

`na.omit (x)` – виключає відсутні значення об'єкта

`na.fail (x)` – повертає виняток, якщо об'єкт містить відсутні значення

`unique (x)` – виключає з об'єкта повторювані елементи

`table (x)` – створює таблицю з кількістю повторень кожного унікального елемента

`subset (x, ...)` – повертає підмножину елементів, що відповідають заданій умові

`sample (x, size)` – повертає випадкову сукупність розміру *size* з елементів *x*

`replace (x, list, values)` – замінює значення *x* з індексами з *list* значеннями з *values*

`append (x, values)` – додає елементи *values* до вектора *x*

## Математика

`sin (x), cos (x), tan (x), asin (x), acos (x), atan (x), atan2 (y, x), log (x), log (x, base), log10 (x), exp (x)` – елементарні математичні функції

`min (x), max (x)` – мінімальний і максимальний елементи об'єкта

`range (x)` – вектор з мінімального і максимального елемента об'єкта

`rmin (x, y), rmax (x, y)` – повертають вектор з мінімальними (максимальними) дляожної пари *x* [*i*], *y* [*i*]

`sum (x)` – сума елементів об'єкта

`prod (x)` – добуток елементів об'єкта

`diff (x)` – повертає вектор різниць між сусідніми елементами

`mean (x)` – середнє арифметичне елементів об'єкта

`median (x)` – медіана об'єкта

`weighted.mean (x, w)` – середньозважене об'єкта *x* (*w* визначає ваги)

`round (x, n)` – округлює *x* до *n* знаків після коми

`cumsum (x)`, `cumprod (x)`, `cummin (x)`, `cummax (x)` – кумулятивні суми, добутки, мініуми та максимуми вектора  $x$  ( $i$ -ий елемент містить статистику за елементами  $x[1:i]$ )

`union (x, y)`, `intersect (x, y)`, `setdiff (x, y)`, `setequal (x, y)`, `is.element (el, set)` – операції над множинами: об'єднання, перетин, різниця, порівняння, принадлежність

`Re (x)`, `Im (x)`, `Mod (x)`, `Arg (x)`, `Conj (x)` – операції над комплексними числами: ціла частина, уявна частина, модуль, аргумент, поєднане число

`fft (x)`, `mvfft (x)` – швидке перетворення Фур'є

`choose (n, k)` – кількість сполучень

`rank (x)` – ранжує елементи об'єкта

## Матриці

`% *%` – матричне множення

`t (x)` – транспонована матриця

`diag (x)` – діагональ матриці

`solve (a, b)` – вирішує систему рівнянь  $a*x = b$

`solve (a)` – зворотна матриця

`colSums`, `rowSums`, `colMeans`, `rowMeans` – суми та середні за стовпцями і за рядками

## Обробка даних

`apply (X, INDEX, FUN = )` – повертає вектор, масив або список значень, отриманих шляхом застосування функції  $FUN$  до певних елементів масиву або матриці  $x$ ; повинні стати предметом обробки елементи  $x$ , вказуються за допомогою аргументу *MARGIN*;

`lapply (x, FUN)` – повертає список тієї ж довжини, що  $x$ ; одночасні значення в новому списку будуть результатом застосування функції  $FUN$  до елементів вихідного об'єкта  $x$

`tapply (x, INDEX, FUN = )` – застосовує функцію  $FUN$  до кожної сукупності значень  $x$ , створеної відповідно до рівнів певного фактора; перелік факторів вказується за допомогою аргументу  $INDEX$

`by (data, INDEX, FUN)` – аналог `tapply ()`, застосовуваний до таблиць даних

`merge (a, b)` – об'єднує дві таблиці даних ( $a$  і  $b$ ) за загальними стовпцями або рядками

`aggregate (x, by, FUN)` – розбиває таблицю даних  $x$  на окремі сукупності даних, застосовує до них певну функцію  $FUN$  і повертає результат в зручному для читання форматі

`stack (x, ...)` – перетворює дані, подані в об'єкті  $x$  у вигляді окремих стовпців, на таблицю даних

`unstack (x, ...)` – виконує операцію, зворотну до дії функції `stack ()`

`reshape (x, ...)` – перетворює таблицю даних з «широкого формату» (повторні вимірювання будь-якої величини записані в окремих стовпцях таблиці) на таблицю «вузького формату» (повторні вимірювання йдуть одне під одним в межах одного стовпчика)

### Рядки (string)

`print (x)` – виводить на екран  $x$

`format (x)` – форматує об'єкт  $x$  так, щоб він виглядав красиво під час виведення на екран

`paste (...)` – конвертує вектори в текстові змінні та об'єднує їх в текстове вираження

`substr (x, start, stop)` – отримання підрядка

`strsplit (x, split)` – розбиває рядок  $x$  на підрядки відповідно до  $split$

`grep (pattern, x)` (а також `grepl`, `regexpr`, `gregexpr`, `regexec`)

– пошук за регулярним виразом

`gsub (pattern, replacement, x)` (а також `sub`) – заміна за

регулярним виразом

`tolower (x)` – звести рядок до нижнього регістру

`toupper (x)` – звести рядок до верхнього регістру

`match (x, table)`, `x % in% table` – виконує пошук елементів у векторі *table*, що збігаються зі значеннями з вектора *x*

`rmatch (x, table)` – виконує пошук елементів у векторі *table*, що частково збігаються з елементами вектора *x*

`nchar (x)` – повертає кількість знаків у рядку *x*

### ***Дата та час***

`as.Date (s)` – конвертує вектор *s* в об'єкт класу *Date*

`as.POSIXct (s)` – конвертує вектор *s* в об'єкт класу *POSIXct*

### **Малювання графіків**

`plot (x)` – графік *x*

`plot (x, y)` – графік залежності *y* від *x*

`hist (x)` – гістограма

`barplot (x)` – столбчаста діаграма

`dotchart (x)` – діаграма Клівленда

`pie (x)` – кругова діаграма

`boxplot (x)` – графік типу «коробочки з вусами»

`sunflowerplot (x, y)` – те саме, що і `plot ()`, проте точки з одинаковими координатами зображуються у вигляді «ромашок», кількість пелюсток у яких пропорційно кількості таких точок

`coplot (x~y | z)` – графік залежності *y* від *x* для кожного інтервалу значень *z*

`interaction.plot (f1, f2, y)` – якщо  $f1$  і  $f2$  – фактори, ця функція створить графік із середніми значеннями у відповідно до значень  $f1$  (по осі  $x$ ) і  $f2$  (по осі  $y$ , різні криві)

`matplot (x, y)` – графік залежності стовпців  $y$  від стовпців  $x$

`fourfoldplot (x)` – зображує (у вигляді частин кола) зв'язок між двома бінарними змінними в різних сукупностях

`assocplot (x)` – графік Кохена-Френдлі

`mosaicplot (x)` – мозаїчний графік залишків лог-лінійної регресії

`pairs (x)` – якщо  $x$  – матриця або таблиця даних, ця функція зобразить діаграми розсіювання для всіх можливих пар змінних з  $x$

`plot.ts (x), ts.plot (x)` – зображує часовий ряд

`qqnorm (x)` – квантилі

`qqplot (x, y)` – графік залежності квантилів  $y$  від квантилів  $x$

`contour (x, y, z)` – виконує інтерполяцію даних і створює контурний графік

`filled.contour (x, y, z)` – те саме, що `contour ()`, але заповнює області між контурами певними кольорами

`image (x, y, z)` – зображує вихідні дані у вигляді квадратів, колір яких визначається значеннями  $x$  і  $y$

`persp (x, y, z)` – те саме, що `image ()`, але у вигляді тривимірного графіка

`stars (x)` – якщо  $x$  – матриця або таблиця даних, зображує графік у вигляді «зірок» так, що кожен рядок являє собою «зірку», а стовпці задають довжину сегментів цих «зірок»

`symbols (x, y, ...)` – зображує різні символи відповідно до координат

`termplot (mod.obj)` – зображує окремі ефекти змінних з регресійної моделі

### **Малювання графіків на низькому рівні**

`points (x, y)` – малювання точок

`lines (x, y)` – малювання лінії

`text (x, y, labels, ...)` – додавання текстового напису

`mtext (text, side = 3, line = 0, ...)` – додавання текстового напису

`segments (x0, y0, x1, y1)` – малювання відрізка

`arrows (x0, y0, x1, y1, angle = 30, code = 2)` – малювання стрілочки

`abline (a, b)` – малювання похилої прямої

`abline (h = y)` – малювання вертикальної прямої

`abline (v = x)` – малювання горизонтальної прямої

`abline (lm.obj)` – малювання регресійної прямої

`rect (x1, y1, x2, y2)` – малювання прямокутника

`polygon (x, y)` – малювання багатокутника

`legend (x, y, legend)` – додавання легенди

`title ()` – додавання заголовка

`axis (side, vect)` – додавання осей

`rug (x)` – малювання зарубок на осі  $X$

`locator (n, type = "n", ...)` – повертає координати на графіку, на які клікнув користувач

### **Lattice-графіка**

`xyplot (y~x)` – графік залежності  $y$  від  $x$

`barchart (y~x)` – стовбчата діаграма

`dotplot (y~x)` – діаграма Клівленда

`densityplot (~x)` – графік щільності розподілу значень  $x$

`histogram (~x)` – гістограма значень  $x$

`bwplot (y~x)` – графік типу «коробочки з вусами»

`qqmath (~x)` – аналог функції `qqnorm()`

`stripplot (y~x)` – аналог функції `stripplot(x)`

`qq (y~x)` – зображує квантилі розподілів  $x$  і  $y$  для візуального порівняння цих розподілів; змінна  $x$  повинна бути числовою, змінна  $y$  – числовою, текстовою або фактором з двома рівнями

`splom (~x)` – матриця діаграм розсіювання (аналог функції `pairs()`)

`levelplot (z~xy | g1g2)` – кольоровий графік значень  $z$ , координати яких задані змінними  $x$  і  $y$  ( $x$ ,  $y$  і  $z$  повинні мати однакову довжину);  $g1$ ,  $g2$  ... (якщо наявні) – фактори або числові змінні, чиї значення автоматично розбиваються на рівномірні відрізки

`wireframe (z~xy | g1g2)` – функція для побудови тривимірних діаграм розсіювання і площин;  $z$ ,  $x$  і  $y$  – числові вектори;  $g1$ ,  $g2$  ... (якщо наявні) – фактори або числові змінні, чиї значення автоматично розбиваються на рівномірні відрізки

`cloud (z~xy | g1g2)` – тривимірна діаграма розсіювання

## Оптимізація та підбір параметрів

`optim (par, fn, method =)` – оптимізація загального призначення

`nlm (f, p)` – мінімізація функції  $f$  алгоритмом Ньютона

`lm (formula)` – підгонка лінійної моделі

`glm (formula, family =)` – підгонка узагальненої лінійної моделі

`nls (formula)` – нелінійний метод найменших квадратів

`approx (x, y =)` – лінійна інтерполяція

`spline (x, y =)` – інтерполяція кубічними сплайнами

`loess (formula)` – підгонка поліноміальної поверхні

`predict (fit, ...)` – побудова прогнозів

`coef (fit)` – розрахункові коефіцієнти

## Статистика

`sd (x)` – стандартне відхилення

`var (x)` – дисперсія

`cor (x)` – кореляційна матриця

`var (x, y)` – коваріація між  $x$  і  $y$

`cor (x, y)` – лінійна кореляція між  $x$  і  $y$

`aov (formula)` – дисперсійний аналіз

`anova (fit, ...)` – дисперсійний аналіз для підгнаних моделей  $fit$

`density (x)` – ядерні щільності ймовірностей

`binom.test ()` – точний тест гіпотези про ймовірність успіху в випробуваннях Бернуллі

`pairwise.t.test ()` – попарні порівняння незалежних або залежних вибірок

`prop.test ()` – перевірка гіпотези про те, що частоти будь-якої ознаки рівні у всіх аналізованих групах

`t.test ()` – тест Стьюдента

## Розподіл

`rnorm (n, mean = 0, sd = 1)` – нормальний розподіл

`rexp (n, rate = 1)` – експоненціальний розподіл

`rgamma (n, shape, scale = 1)` – гамма-розподіл

`rpois (n, lambda)` – розподіл Пуассона

`rweibull (n, shape, scale = 1)` – розподіл Вейбулла

`rcauchy (n, location = 0, scale = 1)` – розподіл Коші

`rbeta (n, shape1, shape2)` – бета-розподіл

`rt (n, df)` – розподіл Стьюдента

`rf (n, df1, df2)` – розподіл Фішера

`rchisq (n, df)` – розподіл Пірсона

`rbinom (n, size, prob)` – біноміальний розподіл  
`rgeom (n, prob)` – геометричний розподіл  
`rhyper (nn, m, n, k)` – гіпергеометричний розподіл  
`rlogis (n, location = 0, scale = 1)` – логістичний розподіл  
`rlnorm (n, meanlog = 0, sdlog = 1)` – логнормальний розподіл  
`runif (n, min = 0, max = 1)` – рівномірний розподіл

## Програмування

### *Робота з функціями*

`function (arglist) {expr}` – створення користувальницької функції  
`return (value)` – повернення значення  
`do.call (funname, args)` – викликає функцію за назвою

### *Умовні оператори*

`if (cond) expr`  
`if (cond) cons.expr else alt.expr`  
`ifelse (test, yes, no)`

### *Цикли:*

`for (var in seq) expr`  
`while (cond) expr`  
`repeat expr`  
`break` – зупинка циклу

## **Додаток 2. Довідкова карта по машинному навчанню**

### **СТАТИСТИКА**

#### *Описова статистика (Summarization)*

Summary ()	узагальнення результатів
summary ()	описові статистичні дані (Hmisc)
boxplot.stats ()	діаграма розмахів (box plot) та інші статистики

#### *Дисперсійний аналіз (Analysis of Variance)*

aov ()	оцінювання та виведення таблиць дисперсійного аналізу
anova ()	розраховує таблицю аналізу дисперсії

#### *Статистичні тести (Statistical Tests)*

chisq.test ()	тест хі-квадрат для таблиць суперечності та оцінки якості підгонки моделей
ks.test ()	тест Колмогорова-Смирнова
t.test ()	тест за t-критерієм Стьюдента
prop.test ()	тест на значущість заданої пропорції
binom.test ()	точний біноміальний тест

#### *Регресійний аналіз (Regression Functions)*

lm ()	лінійні моделі
glm ()	узагальнені лінійні моделі
gbm ()	узагальнені регресійні бустінг-моделі (gbm)
predict()	метод для отримання розрахованих значень
residuals()	залишки моделей, різниця фактичних та розрахункових значень
nls ()	нелінійна регресія
gls (), gnls ()	побудова лінійних та нелінійних моделей методом узагальнених найменших квадратів (nlme)

*Моделі змішаних ефектів (nlme) (Mixed Effects Models)*

lme (), nlme () лінійні та нелінійні моделі із змішаними ефектами

*Факторний аналіз (Principal Components and Factor Analysis)*

princomp (),

prcomp () аналіз головних компонент і факторний аналіз

*Пошук викидів (Outlier Detection)*

boxplot.stats ()

\$ out перелік спостережень, що знаходяться за межами інтервалу, що обмежений «вусами» діаграми розмахів

lofactor () розрахунок фактору локальних викидів за LOF-алгоритмом (DMwR або dprep)

lof () паралельна реалізація LOF-алгоритму (Rlof)

*Інші функції*

var (), cov (),

cor () дисперсія, коваріація, кореляція

density() ядерна щільність

cmdscale () багатовимірне шкалювання (MDS)

*Пакети*

gbm узагальнені регресійні бустінг-моделі

nlme лінійні та нелінійні моделі зі змішаними ефектами

Rlof паралельна реалізація LOF-алгоритму

extremevalues пошук екстремальних значень у одновимірних даних

outliers загальні методи пошуку викидів

mvoutlier пошук багатовимірних викидів із застосуванням

робастних методів

## ЧАСОВІ РЯДИ

### *Перетворення та відображення (Construction & Plot)*

ts ()	створення об'єктів класу «часовий ряд»
plot.ts ()	метод для візуалізації об'єктів класу «часовий ряд»
smoothts()	згладжування тимчасових рядів (ast)
sfilter ()	видалення сезонних флюктуацій із застосуванням ковзаючого середнього (ast)

### *Декомпозиція (Decomposition)*

decomp ()	декомпозиція часового ряду за фільтром квадратного кореня (timsac)
decompose ()	класична сезонна декомпозиція за ковзаючим середнім
stl ()	сезонна декомпозиція часового ряду за локальною регресією
tsr ()	декомпозиція часового ряду (ast)
ardec ()	декомпозиція часового ряду за авторегресіями (ArDec)

### *Прогнозування (Forecasting)*

arima ()	модель ARIMA для одновимірного часового ряду
predict.Arima ()	прогноз за моделями ARIMA
auto.arima ()	підгонка оптимальної моделі ARIMA для одновимірного ряду (forecast)
forecast.stl (), forecast.ets (),	
c. Arima ()	прогноз з використанням STL, ETS та ARIMA моделей

### *Кореляція та коваріація (Correlation and Covariance)*

acf ()	автокореляція та автоковаріація часового ряду
ccf ()	крос-коваріація та крос-кореляція двох одновимірних часових рядів

*Пакети*

forecast	аналіз та візуалізація прогнозу одновимірного часового ряду
hts	аналіз та прогнозування ієрархічних та згрупованих рядів
TSclust	функції для кластеризації часових рядів
dtw	динамічна трансформація шкали часу (Dynamic Time Warping, DTW)
timsac	функція для аналізу та перетворення часових рядів
ast	аналіз часових рядів
ArDec	декомпозиція часового ряду, основана на авторегресії
dse	засоби для створення багатовимірних, лінійних та інваріантних моделей часових рядів

## АССОЦІАТИВНІ ПРАВИЛА

apriori ()	пошук асоціацій за алгоритмом Apriori, що розраховує частоту подій, що відбуваються одночасно, априорі відкидаючи маломовірні (arules)
eclat ()	пошук сукупностей подій за алгоритмом Eclat, що перебирає класи еквівалентності та їх переселення (arules)
cspade ()	пошук частих фрагментів послідовностей за алгоритмом cSPADE (arulesSequences)
seqefsub ()	пошук за частими підпослідовністями (TraMineR)

*Пакети*

arules	пошук частих, максимальних або закритих сукупностей подій та асоціативних правил з використанням алгоритмів Apriori та Eclat
arulesViz	візуалізація асоціативних правил

arulesSequences дополнение к arules для обробки і виділення частих послідовностей

TraMineR пошук, опис та візуалізація послідовностей об'єктів або подій

## КЛАССИФІКАЦІЯ ТА ПРОГНОЗ

### *Дерева рішень (Decision Trees)*

ctree () дерево умовного виводу, рекурсивне розбиття для постійних, впорядкованих, категоріальних і багатовимірних відкликів у рамках умовного виводу (party)

rpart () дерево рекурсивного розбиття та регресії (rpart)

mob () рекурсивне розбиття, що приводить до створення дерев, вузли яких містять статистичні моделі для відповідних сукупностей даних (party)

varimp () важливість предикторів (party)

### *Випадковий ліс (Random Forest)*

cforest () ансамблі моделей, створені з використанням алгоритмів «випадний ліс» і «бэггинг» (party)

randomForest () випадковий ліс (randomForest)

importance() важливість предикторів (randomForest)

### *Нейронні мережі (Neural Networks)*

nnet () будує нейронну мережу з одним прихованим шаром (nnet)

neuralnet () навчання нейронних мереж (neuralnet)

mlp (), dlvq (), rbf (),

rbfDDA (), elman (),

jordan (), som (), art1 (), art2 (),  
artmap (), assoz () типи нейронних мереж (RSNNS)

*Машини опорних векторів (Support Vector Machine – SVM)*

svm ()	навчання машини опорних векторів для регресії, класифікація або оцінка щільності вірогідності (e1071)
ksvm ()	машини ядерних опорних векторів (kernlab)

*Байєсовські класифікатори (Bayes Classifiers)*

naiveBayes ()	наївний байєсовський класифікатор (e1071)
---------------	---

*Оцінка ефективності моделі (Performance Evaluation)*

performance ()	розраховує різні показники якості моделей (ROCR)
PRcurve ()	криві чутливості та специфічності класифікатора (DMwR)
CRchart ()	графік кумулятивної чутливості класифікатора (DMwR)
roc ()	побудова ROC-кривої (pROC)
auc ()	обчислення площин під ROC-кривою (pROC)
ROC ()	візуалізація ROC-кривої (DiagnosisMed)

*Пакети*

party	рекурсивне розбиття
rpart	дерева рекурсивного розбиття та регресії
rpartOrdinal	дерева класифікації для даних з впорядкованими категоріями
rpart.plot	метод візуалізації rpart-моделей
randomForest	класифікація та регресія на основі випадкового лісу
caret	моделі класифікації та регресії
nnet	нейронні мережі зустрічного розповсюдження

RSNNS	Штутгартський Симулятор нейронних мереж
neuralnet	нейронні мережі зворотного розповсюдження
e1071	функції для аналізу латентних класів, Фур'є-перетворення коротких часових рядів, нечіткої кластеризації, машин опорних векторів, обчислення найкоротшого шляху, бэггінг-кластеризація, найвна байесова класифікація тощо
ROCR	візуалізації якості класифікаторів
pROC	візуалізація та аналіз ROC-кривих

## КЛАСТЕРИЗАЦІЯ

*Кластеризація, основана на розбитті (Partitioning based Clustering)*

Розділення даних на k груп з наступним експериментальним покращенням якості кластеризації шляхом переміщення об'єктів з однієї групи до іншої

kmeans ()	виконує кластеризацію за методом k середніх
kmeansruns ()	виводить функцію kmeans () для кластеризації за методом k середніх і знаходження оптимальної кількості кластерів на основі декілька початкових розміщень їх центрів (fpc)
kmeansCBI ()	інтерфейс для взаємодії з функцією з пакетів kmeans (fpc)
cluster.optimal ()	пошук найкращої кулькості кластерів у деякій сукупності даних (bayesclust)
clara ()	кластеризація для великих сукупностей даних (cluster)
fanny (x, k,...)	нечітка кластеризація з k кластерами (cluster)
kcca ()	кластеризація за k центроїдами (flexclust)
ccfkms ()	кластеризація із застосуванням кон'югатних опуклих функцій (cva)

apcluster ()	кластеризація за методом «передачі повідомень» (affinity propagation) на основі вхідної матриці подібностей (apcluster)
apclusterK ()	кластеризація за методом «передачі повідомень» для знаходження k кластерів (apcluster)
cclust ()	опукла кластеризація включно із методом k середніх і двох інших методів (cclust)
KMeansSparseCluster ()	кластеризація за методом k середніх із одночасним знаходженням інформаційних змінних (sparcl)
tclust (x, k, alpha,...)	кластеризація за методом k усічених середніх (частина даних видаляється з розгляду) (tclust)

### *Ієрархічна кластеризація (Hierarchical Clustering)*

Ієрархічна декомпозиції даних або знизу догори (агломерація) або згори донизу (розділення)

hclust ()	ієрархічний кластерний аналіз за матрицею відстаней
birch()	алгоритм Birch для кластеризації великих даних з використанням CF-дерев (birch)
pvclust ()	ієрархічна кластеризація з одночасним обчисленням p-значень на основі бутстреп-вибірок різного розміру (pvclust)
agnes ()	агломеративний ієрархічний кластерний аналіз (cluster)
diana ()	ієрархічний кластерний аналіз на основі розділення (cluster)
mona ()	ієрархічний кластерний аналіз на основі розділення для даних, поданих лише бінарними змінними (cluster)
rockCluster ()	кластеризація з використанням алгоритму Rock (cba)
proximus ()	кластеризація на основі алгоритму Proximus для даних, що подані тільки бінарними змінними (cba)

isopam()	алгоритм кластеризації Isopam (isopam)
flashClust ()	оптимальна ієрархічна кластеризація (flashClust)
fastcluster ()	швидка ієрархічна кластеризація (fastcluster)
cutreeDynamic (),	
cutreeHybrid ()	виділення кластерів на дендрограмах (dynamicTreeCut)
Hierarchical	
SparseCluster ()	ієрархічна кластеризація з одночасним знаходженням інформативних змінних (sparcl)

*Моделі, основані на кластерах (Model based Clustering)*

Mclust ()	кластеризація на основі статистичних моделей (mclust)
HDDC ()	кластеризація на основі статистичних моделей для великих даних (HDclassif)
fixmahal ()	кластеризація із застосуванням фіксованих точок і відстані Махalanобіса (fpc)
fixreg ()	кластеризація на основі регресії із застосуванням фіксованих точок (fpc)
mergenormals ()	кластеризація, заснована на злитті компонент змішаного гауссового розподілу (fpc)

*Кластеризація, основана на щільності (Density based Clustering)*

Формування кластерів шляхом об'єднання ділянок з щільним розміщенням точок

dbscan (data, eps, MinPts,...)	знаходження кластерів довільної форми з використанням параметрів eps (радіус, в межах якого перебувають точки-сусіди) і MinPts (порогове значення щільності розташування точок) (fpc)
-----------------------------------	---

pdfCluster ()      кластеризація на основі ядерної щільності ймовірності  
(pdfCluster)

*Інші методи кластеризації (Other Clustering Techniques)*

mixer()      кластеризація на основі випадкових графів (mixer)  
nncluster ()      швидка кластеризація на основі алгоритму restarted  
minimum spanning tree (nnclust)  
orclus ()      кластеризація на основі алгоритму ORCLUS (orclus)

*Відображення результатів кластеризації (Plotting Clustering Solutions)*

plotcluster ()      візуалізація груп даних (fpc)  
bannerplot ()      горизонтально орієнтований «бітлот», зображаючий  
результат ієрархічної кластеризації (cluster)

*Оцінка якості кластеризації (Cluster Validation)*

silhouette ()      надає інформацію за «силуетами» кластерів (cluster)  
cluster.stats ()      обчислює статистичні показники якості кластеризації на  
основі матриці відстаней (fpc)  
c1Valid ()      обчислює статистику якості для декількох алгоритмів  
кластеризації та різної кількості кластерів (c1Valid)  
clustIndex ()      обчислює індекси кластеризації, що можна  
використовувати для знаходження оптимальної кількості  
кластерів (cclust)  
NbClust ()      дозволяє вивести 30 індексів для оцінки якості  
кластеризації та знаходження оптимальної кількості  
кластерів (NbClust)

*Пакети*

Cluster      кластерний аналіз

fpc	різні методи кластеризації та валідації отриманих рішень
mclust	кластеризація, заснована на статистичних моделях
birch	кластеризація великих сукупностей даних з використанням алгоритму Birch
pvclust	ієрархічна кластеризація з розрахунком p-значень
apcluster	кластеризація на основі методу «передачі повідомлень»
cclust	методи опуклої кластеризації включно із алгоритмом k-середніх, а також обчислення індексів для знаходження оптимальної кількості кластерів
cba	кластеризація для вирішення бізнес-задач включно із такими алгоритмами, як Proximus і Rock
bclust	байесівська кластеризація на основі ієрархічної моделі, для великих даних
biclust	алгоритми для нахождень двовимірних кластерів
clue	ансамблі кластерних рішень
clues	метод кластеризації, заснований на локальній регуляризації
c1Valid	валідація кластерних рішень
clv	методи валідації кластерних рішень включно із методами внутрішньої і зовнішньої валідації
clustsig	аналіз статистичної значущості кластерів, а також різниці між кластерами
clusterSim	пошук найпростішої процедури кластеризації
clusterGeneration	імітація кластерів
gcExplorer	інструмент для графічного аналізу результатів кластеризації
hybridHclust	гібридний ієрархічний кластерний аналіз на основі «сумісних кластерів»

Modalclust	ієрархічний кластерний аналіз на основі мод
EMCC	еволюційні методи Монте-Карло (ЕМС) для кластеризації
rEMM	модель Маркова (ЕММ) для кластеризації потокових даних

## ОБРОБКА ТЕКСТІВ

Імпорт, очищення та підготовка текстів ((Importing Text, Cleaning and Preparation)

readPDF ()	вилення тексту та метаданих з документів формату PDF (tm)
Corpus ()	створення корпуса, тобто колекції з декількох документів (tm)
tm_map ()	перетворення текстових документів, тобто стеммінг, видалення стоп-слів тощо (tm)
tm_filter ()	фільтрування документів з корпусу (tm)
TermDocumentMatrix (), DocumentTermMatrix ()	створення терм-документальних і документо-термних матриць (tm)
Dictionary ()	створення словника з текстового вектору або терм-документної матриці (tm)
stemDocument ()	стемінг слів в документах (tm)
stemCompletion ()	відтворення повної форми слів після стеммінга (tm)
SnowballStemmer ()	стемінг за алгоритмом Snowball (Snowball)
stopwords(language)	стоп-слова з різних мов (tm)
removeNumbers (), RemovePunctuation (), RemoveWords ()	видалення чисел, знаків пунктуації або сукупності слів з документа (tm)

deleteSparseTerms () видалення рідких термів із терм-документальної матриці (tm)

*Часті терміни та асоціації (Frequent Terms and Association)*

findAssocs () знаходить зв'язок між термами в терм-документальних матрицях (tm)

findFreqTerms () знаходить часті терміни в терм-документальних матрицях (tm)

termFreq () формує вектор з частотами термів для заданого документа (tm)

*Тематичне моделювання (Topic Modelling)*

LDA () підгонка LDA-моделі (Latent Dirichlet Allocation)  
(topicmodels)

CTM () підгонка CTM-моделі (Correlated Topics Model)  
(topicmodels)

terms () витяг найбільш вірогідних термів для заданої теми  
(topicmodels)

topics () витяг найбільш вірогідних тем для заданого документа  
(topicmodels)

polarity () індекс полярності (в аналізі тональності текстів) (qdap)

textcat () класифікація текстів на основі n-грам (textcat)

*Візуалізація тексту (Text Visualization)*

wordcloud () створення «хмари слів» (wordcloud)

comparison.cloud() створює «хмари слів» для порівняння частоти зустрічності цих слів у різних документах (wordcloud)

commonality.cloud () «хмари слів» загальних для декількох документів (wordcloud)

*Пакети*

tm	сукупність утиліт для аналізу текстів
topicmodels	підгонка LDA- та СТМ-моделей
wordcloud	створення «хмари слів»
lda	підгонка LDA-моделей
Wordnet R	інтерфейс для лексичної бази даних WordNet
RTextTools	автоматична класифікація документів шляхом навчання з учителем
qdap	сукупність утиліт для аналізу природної мови та документів
sentiment140	аналіз тональності текстів із використанням безкоштовного сервісу sentiment140
tm.plugin.dc	додатковий модуль для пакетів tm, дозволяє організовувати розподілені обчислення під час аналізу текстів
tm.plugin.mail	додатковий модуль для пакетів tm, що полегшує роботу з текстами електронної пошти
textir	сукупність утиліт для виконання аналізу тональності тексту та оцінювання статистичних різниць між документами
tau	сукупність утиліт для аналізу текстів

## АНАЛІЗ СОЦІАЛЬНИХ МЕРЕЖ ТА ФУНКЦІЇ РОБОТИ З ГРАФАМИ

graph (),  
graph.edgelist (),  
graph.adjacency (),  
graph.incidence () створення графів на основі таких структур, як «ребра» (edges), «список ребер» (edge list), «матриця відстаней»

(adjacency matrix) і «матриця частотності» (incidence matrix) (igraph)

plot (), tkplot (),

rglplot () статичні, інтерактивні та трьохвимірні зображення графів (igraph)

gplot (), gplot3d () візуалізація графів (sna)

vcount (),

ecount () кількість ребер та вузлів (igraph)

V (), E () доступ до вузлів і ребер графа (igraph)

is.directed () чи є граф спрямованим (igraph)

are.connected () чи пов'язані два вузли графа? (igraph)

degree (), betweenness (),

closeness (), closeness (),

evcent () міри центральності графа (igraph, sna)

edge.density () щільність графа (igraph)

add.edges (),

add.vertices (),

delete.edges (),

delete.vertices () додавання та видалення вузлів та ребер (igraph)

neighborhood () знаходження сусідів заданого вузла графа (igraph, sna)

get.adjlist () отримання списків сусідніх вузлів або ребер (igraph)

nei (), adj (), from (),

to () індексування вузлів і ребер графа (igraph)

cliques (), large.cliques (),

maximal.cliques (),

clique.number () знаходження повних підграфів (igraph)

clusters (),

no.clusters () знаходження максимально пов'язаних елементів графа і їх кількості (igraph)

fastgreedy.community ()	
spinglass.community ()	алгоритми знаходження повідомлень у графах (igraph)
cohesive.blocks ()	знаходження кластерів в графах (igraph)
induced.subgraph ()	вивільнення частин графа (igraph)
mst ()	алгоритм мінімального оственного дерева (igraph)
components ()	знаходження максимально пов'язаних компонентів графа (igraph)
shorttest_paths ()	знаходження найкоротшого шляху між вузлами (igraph)
% ->% , % <-%, % -%	оператори індексування ребер графа (igraph)
get.edgelist ()	повертає список ребер у вигляді матриці з двома стовбцями (igraph)
read.graph (), write.graph ()	імпорт і експорт графів у вигляді файлів різних форматів (igraph)

### *Пакети*

igraph	аналіз та візуалізація графів
sna	аналіз соціальних мереж
d3Network,	
networkD3	R-інтерфейс к JavaScript-бібліотекам D3 для побудови графів, дерев, дендрограм і діаграм Санки
RNeo4j	взаємодія з базами даних Neo4j з R
statnet	сукупність інструментів для опису, візуалізації, аналізу та імітації графів
egonet	виміри центральності для аналізу соціальних мереж
network	інструменти для створення та модифікації графів
bipartite	візуалізація графів та обчислення статистик

blockmodeling	узагальнення і моделювання блоків в розмічених графах
blockmodeling	візуалізація простих графів, побудова потокових діаграм
NetCluster	кластеризація елементів графа
NetData	дані для лабораторних робіт з аналізу соціальних мереж із підтримкою R від McFarland тощо
NetIndices	розраховує різні мережеві індекси
NetworkAnalysis	оцінка статистичних різниць між графами
tnet	аналіз бімодальних та динамічних графів

### ФУНКЦІЇ ДЛЯ РОБОТИ З ПРОСТОРОВИМИ ДАНИМИ

Geocode ()	геокодування із застосуванням сервісу Google Maps (ggmap)
plotGoogleMaps ()	візуалізація даних на Google Maps (plot-GoogleMaps)
qmap ()	побудова карт (ggmap)
get_map ()	запрос до служби Google Maps, OpenStreetMap або Stamen Maps для побудови карт (ggmap)
gvisGeoChart (),	
gvisGeoMap (),	
gvisIntensityMap (),	
gvisMap ()	гео-діаграми та карти від Google (googleVis)
GetMap ()	завантажує статичну карту з сервера Google (RgoogleMaps)
ColorMap ()	кольорове кодування та зображення значення змінної на карті (RgoogleMaps)
PlotOnStaticMap ()	суміщення графіків із географічними картами (RgoogleMaps)
TextOnStaticMap ()	текстові мітки на карти (RgoogleMaps)

*Пакети*

plotGoogleMaps	візуалізація даних на карті Google Maps і збереження результатів у вигляді HTML-віджетів
RgoogleMaps	робота з картами Google Maps в R
ggmap	візуалізація даних із використанням сервісів Google Maps та OpenStreetMap
plotKML	візуалізація просторових і просторово-часових об'єктів з використанням Google Earth
SGCS	кластеризація геоінформаційних даних із використанням просторових графів
spdep	інструменти для пошуку просторів залежностей

**ГРАФІЧНІ ФУНКЦІЇ**

plot ()	функція загального призначення для візуалізації даних
barplot (), pie (), hist ()	стовпчикові діаграми, кругові діаграми і гістограми
boxplot ()	діаграми розмахів
stripchart ()	одновимірні діаграми розсіювання
dotchart ()	точкова діаграма Клівенда
qqnorm (),	
qqplot (), qqline ()	графіки квантіль-квантіль
coplot ()	категоризовані графіки
splom ()	категоризовані матриці діаграм розсіювання (lattice)
pairs ()	матриці діаграм розсіювання
cpairs ()	поліпшенні матриці діаграм розсіювання (gclus)
parcoord ()	діаграма паралельних координат (MASS)
sparcoord ()	поліпшенні діаграми паралельних координат (gclus)
paralplot ()	діаграма паралельних координат (lattice)
щільність ()	графік ядерної функції щільності (lattice)
contour (),	

fill.contour ()	контурні діаграми
levelplot (),	
contourplot ()	контурні діаграми (lattice)
mosaicplot ()	мозаїчна діаграма
assocplot ()	діаграма асоціацій
smoothScatter ()	діаграма розсіювання з кольоровим поданням щільності ймовірності; для візуалізації великих масивів даних
sunflowerplot ()	діаграма типу «соняшник»
matplot ()	зображення стовпців однієї матриці в залежності від стовпців іншої матриці
fourfoldplot ()	візуалізація таблиць спряженості розміру $2 \times 2 \times k$
persp ()	трьохвимірні діаграми поверхонь
cloud (), wireframe ()	трьохвимірні діаграми розсіювання та діаграми поверхонь (lattice)
interaction.plot ()	графік взаємодій між двома змінними
iplot (), ihist (),	
ibar (), ipcp ()	інтерактивні діаграми розсіювання, гістограми, стовпчикові діаграми та діаграми паралельних координат (iplots)
pdf (), postscript (),	
win.metafile (),	
jpeg (), bmp (),	
png (), tiff ()	збереження графіків у файлах різних форматів
gvisAnnotatedTimeLine (), gvisAreaChart (), gvisBarChart (),	
gvisBubbleChart (), gvisCandlestickChart (), gvisColumnChart (),	
gvisComboChart (), gvisGauge (), gvisGeoChart (), gvisGeoMap (),	
gvisIntensityMap (), gvisLineChart (), gvisMap (), gvisMerge (),	
gvisMotionChart (), gvisOrgChart (), gvisPieChart (),	

gvisScatterChart (), gvisSteppedAreaChart (), gvisTable (),  
 gvisTreeMap ()    різні інтерактивні діаграми, створені з використанням  
                     Google Visualisation API (googleVis)

*Пакети*

ggplot2	реалізація принципів «граматики графічних елементів»
ggvis	інтерактивний варіант реалізації принципів «граматики графічних елементів»
googleVis	інтерфейс між R та Google Visualisation API для створення інтерактивних діаграм
d3Network,	
networkD3	R-інтерфейс до JavaScript-бібліотеки D3 для побудови графів, дерев, дендрограм і діаграм Санки
rCharts	інтерактивні діаграми з використанням різних JavaScript-бібліотек
lattice	високорівнева система візуалізації даних
vcd	візуалізація категорійних даних
iplots	інтерактивні діаграми

**ПЕРЕТВОРЕННЯ ДАНИХ**

transform ()	перетворення таблиць даних
scale ()	нормування матриць і таблиць
t ()	транспонування матриць
aperm ()	транспонування масивів
table (), tabulate (),	
xtabs ()	формування зведеніх таблиць
stack (), untack ()	об'єднання та декомпозиція векторів
split (),	
unsplit ()	розділення даних за групами відповідно до рівня

	деякого фактору (-ів) і зворотна операція
reshape ()	перетворення таблиць даних у «широкий» або «довгий» формат
merge ()	злиття двох таблиць даних
aggregate ()	зведені статистичні дані для окремих груп даних
by ()	застосуванням довільної функції до таблиці даних, що розбито на групи відповідно до рівня деякого фактору
melt (), cast ()	розділяє таблиці даних на складові елементи з формуванням нової таблиці іншої форми і/або вмісту (reshape)
sample ()	формування випадкових вибірок
complete.cases ()	пошук записів у таблиці даних, що не містять пропущених значень
na.fail, na.omit,	
na.exclude,	
na.pass	обробка пропущених значень

*Пакети*

dplyr	високоефективна сукупність утиліт із стандартизованим синтаксисом для роботи з таблицями даних
reshape	гнучкий інструмент для зміни форм таблиць даних і їх агрегування
reshape2	удосконалення пакету reshape
tidyverse	удосконалення пакету reshape2; дозволяє змінювати форму таблиць даних за допомогою функцій spread () та gather ()
data.table	сукупність утиліт для високоефективної роботи з таблицями даних (індексування, join-операції зі збереженням порядку, групування тощо)

gdata	різні програми для маніпуляцій із даними
lubridate	сукупність функцій для роботи з датами та часом
stringr	сукупність функцій для роботи із символічними даними

## ФУНКЦІЇ ДОСТУПУ

save (), load ()	збереження та завантаження об'єктів типу RData
read.csv (),	
write.csv ()	імпорт і експорт файлів форми .csv
read.table (), write.table (),	
scan (), write ()	імпорт і експорт даних
read.xlsx (),	
write.xlsx ()	імпорт і експорт Excel-файлів (xlsx)
read.fwf ()	імпорт даних, що зберігаються у вигляді файлів із фіксованою ширинкою полів
write.matrix ()	експорт матриці або таблиці даних (MASS)
readLines (),	
writeLines ()	імпорт і експорт текстових строк з файлу
sqlQuery ()	виконання SQL-запитів до бази даних ODBC (RODBC)
sqlFetch ()	імпорт таблиці з баз даних ODBC (RODBC)
sqlSave (),	
sqlUpdate ()	збереження та оновлення таблиць у базах даних ODBC (RODBC)
sqlColumns ()	структура таблиці у базі даних (RODBC)
sqlTables ()	спісок таблиць, що знаходяться в базі даних (RODBC)
odbcConnect (), odbcClose (),	
odbcCloseAll ()	відкриття та закриття з'єднання з базою даних ODBC (RODBC)
dbSendQuery	виконання SQL-запиту до бази даних (DBI)
dbConnect (),	

Кононова К. Ю.

dbDisconnect () відкриття та закриття з'єднання із системою управління базами даних (DBI)

*Пакети*

RODBC	доступ до бази даних ODBC
foreign	імпорт і експорт даних в інших форматах, таких як Minitab, S, SAS, SPSS, Stata, Systat тощо
sqldf	виконання SQL-подібних SELECT-запитів до таблиць R
DBI	DBI-інтерфейс між R і реляційними СУБД
RMySQL	драйвер для з'єднання з базами даних MySQL
RJDBC	доступ до бази даних через інтерфейс JDBC
RSSQLite	драйвер для з'єднання з базами даних RSQLite
ROracle	DBI-драйвер для з'єднання з базами даних Oracle
Rpgsql	DBI/RJDBC-інтерфейс для роботи з базами даних PostgreSQL
RODM	інтерфейс для роботи з Oracle Data Mining
xlsx	імпорт і експорт файлів у форматі Excel 97/2000/XP/2003/2007
xlsReadWrite	імпорт і експорт Excel-файлів
WriteXLS	створення файлів Excel 2003 (xls) з таблиць даних R
SPARQL	SPARQL-драйвер для виконання запитів SELECT та UPDATE

**ФУНКЦІЇ ДОСТУПУ ЧЕРЕЗ ВЕБ-ІНТЕРФЕЙС**

download.file () загрузка файлів з інтернету  
xmlParse (),  
htmlParse () розбір файлів XML і HTML (XML)  
userTimeline (), homeTimeline (),  
mentions (),

retweetsOfMe ()	вилучення різноманітних даних з мережі Twitter (twitteR)
searchTwitter ()	пошук у мережі Twitter (twitteR)
getUser (),	
lookupUsers()	інформація про користувачів мережі Twitter (twitteR)
getFollowers (), getFollowerIDs (),	
getFriends (),	
getFriendID ()	спісок фоловерів та друзів того чи іншого користувача мережі Twitter (twitteR)
twListToDF ()	конвертування списків twitteR в стандартні таблиці даних (twitteR)

*Пакети*

twitteR	сукупність інструментів для роботи з Twitter API
RCurl	R-клієнт для виконання запитів за стандартними протоколами (HTTP/FTP/...)
XML	читання та створення документів у форматах XML та HTML
https	набір для роботи з URL та HTTP (побудований на основі RCurl)

**ФУНКЦІЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ**

mapreduce ()	специфікація та виконання завдань MapReduce (rmr2)
keyval ()	створення об'єктів типу «ключ – значення» (rmr2)
from.dfs (),	
to.dfs ()	імпорт і експорт об'єктів R під час роботи з іншими файловими системами (rmr2)
hb.get (), hb.scan (),	
hb.get.data.frame ()	читання таблиць HBase (rbase)
hb.insert (),	

Кононова К. Ю.

hb.insert.data.frame () запис таблиць HBase (rbase)

hb.delete () видалення запису з таблиці HBase (rbase)

### *Пакети*

rmr2	аналіз даних в R в стилі MapReduce на Hadoop-кластері
rdfs	з'єднання з розподіленою файловою системою Hadoop (HDFS)
Rhbase	з'єднання з NoSQL базою даних HBase
Rhipe	інструменти для роботи з Hadoop з R
SparkR	тонкий R-клієнт для роботи з Apache Spark
RHive	розділені обчислення на основі запитів до HIVE
Segue	виконання паралельних обчислень з використанням Amazon Elastic Map Reduce (EMR)
HadoopStreaming	утиліти для використання R-скриптів під час обробки потокових даних на Hadoop-кластері
hive	розділені обчислення, основані на MapReduce
rHadoopClient	R-клієнт для роботи Hadoop

### ВЕЛИКІ ДАНІ

as.ffdf ()	перетворення таблиць даних на формат ffdf (ff)
read.table.ffdf (),	
read.csv.ffdf ()	читання даних із текстового файлу та збереження у ffdf-об'єкті (ff)
write.table.ffdf (),	
write.csv.ffdf ()	збереження ffdf-об'єктів у вигляді текстових файлів (ff)
ffdfappend ()	додавання звичайних таблиць даних або таблиць ffdf до існуючої таблиці ffdf (ff)
big.matrix ()	створення стандартної «великої матриці» (об'єкт типу big.matrix), розміром, що обмежений доступним

об'ємом оперативної пам'яті (**bigmemory**)

**read.big.matrix ()** створює «великі матриці» шляхом читання з ASCII-файлу (**bigmemory**)

**write.big.matrix ()** запис «великої матриці» у файл (**bigmemory**)  
filebacked.

**big.matrix ()** створення «великої матриці» у вигляді файлу на диску (розмір матриці може перевищувати доступний об'єм пам'яті) (**bigmemory**)

**mwhich ()** уdosконалені which-подобні команди для роботи з великими матрицями (**bigmemory**)

### *Пакети*

**ff** зберігання великих масивів даних на диск

**ffbase** стандартні статистичні функції для пакетів ff

**filehash** база даних типу «ключ – значення» для роботи з великими масивами даних

**g.data** створює та підтримує пакети для роботи з даними типу delayed data

**BufferedMatrix** об'єкти для зберігання матриць із даними у тимчасових файлах

**biglm** регресійний аналіз для даних, що не вміщуються в пам'ять комп'ютера

**bigmemory** набір для роботи з матрицями даних великого розміру  
**biganalytics** розширення пакетів **bigmemory**, що містить додаткова сукупність аналітичних функцій

**bigtabulate** table-, tapply- та split-подібні функції для роботи з об'єктами matrix i big.matrix

## ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ

Функції для організацій паралельних обчислень

sfInit (),

sfStop () запуск і завершення роботи кластера (snowfall)

sfLapply (), sfSapply (),

sfApply () паралельні версії функцій lapply (), sapply (), apply()  
(snowfall)

foreach(...)%dopar% паралельне виконання циклу (foreach)

registerDoSEQ (), registerDoSNOW (),

registerDoMC () реєстрація послідовного, SNOW та багатопоточного  
бэк-енда для виконання паралельних обчислень з  
підтримкою пакетів foreach (foreach, doSNOW, doMC)

### *Пакети*

snowfall	«обортка» на основі функціоналу пакета snow для більш ефективної розробки програмного забезпечення для паралельних обчислень у R
snow	організація паралельних обчислень в R
multicore	паралельне виконання R-кода на машинах із кількома процесорами
snowFT	розширення пакетів snow для розробки робастних прикладних програм та для зручної організації паралельних обчислень
Rmpi	інтерфейс для роботи з MPI (Message-Passing Interface)
rpvml	R-інтерфейс для роботи з PVM (Parallel Virtual Machine)
nws	набір інструментів для координації паралельних обчислень
foreach	конструктори foreach-циклів для R
doMC	адаптер до пакету multicore для паралельних обчислень

doSNOW	адаптер до пакетів snow для виконання паралельних foreach-обчислень
doMPI	адаптер до пакету Rmpi для виконання паралельних foreach-обчислень
doParallel	адаптер до пакету multicore для виконання паралельних foreach-обчислень
doRNG	генератор випадкових чисел, що дозволяє виконувати паралельні обчислення на основі foreach-циклів
GridR	виконання R-кода на віддалених машинах і кластерах
fork	набір функцій для одночасної роботи з декількома процесами R

## ІНТЕРФЕЙС ДО ІНШИХ МОВ

.jcall ()	виклик методу Java (rJava)
.jnew ()	створення нового об'єкту Java (rJava)
.jinit ()	ініціалізація віртуальної машини Java (JVM) (rJava)
.jaddClassPath ()	додає JAR-файли до класу (rJava)

### *Пакети*

rJava	низькорівневий інтерфейс між R і Java
rPython	виклик функції Python з R

## ФУНКЦІЇ ДЛЯ ГЕНЕРАЦІЇ ЗВІТІВ

Sweave ()	поєднання тексту з R або S-кодом для автоматичного електронного формування звітів
xtable ()	експорт таблиць у форматах LaTeX або HTML (xtable)

### *Пакети*

knitr	пакет для формування динамічних звітів в R
-------	--

xtable	експорт таблиць у форматах LaTeX або HTML
R2HTML	створення HTML- звітів
R2PPT	формування презентацій Microsoft PowerPoint
RPMG	графічний інтерфейс користувача (GUI) для інтерактивних R-сесій
Red-R	графічний інтерфейс користувача з відкритим кодом для візуального програмування на R
rattle	графічний інтерфейс користувача для Data Mining на R
latticist	графічний інтерфейс користувача для виконання візуального аналізу даних

*Створення графічних інтерфейсів користувача*

shiny	фреймворк для розробки веб-програм в R
svDialogs	створює діалогові вікна
gWidgets	платформонезалежний набір інструментів для розробки графічних інтерфейсів користувача
R AnalyticFlow	виконання аналізу даних за допомогою блок-схем

*Редактори для розробки коду на R*

RStudio	безкоштовна інтегрована середа розробки (IDE) для R
Tinn-R	безкоштовний графічний інтерфейс користувача для R
Rpad	веб-інтерфейс для R у вигляді робочих книг

### **Додаток 3. Пакет `dplyr`**

Dplyr пропонує п'ять основних операцій для маніпуляції даними, що можуть бути застосовані до окремої таблиці: filter (), arrange (), select (), mutate (), summarise () .

#### **Фільтрація рядків за допомогою filter ()**

Filter () дозволяє вибрати підмножину рядків з data.frame. Перший аргумент – назва сукупності даних, другий і наступні – умови фільтра в контексті цієї сукупності даних.

Filter () працює аналогічно subset () за тим винятком, що ви можете йому передати будь-яку кількість умов для фільтра, що будуть об'єднані разом через & (логічне I). Ви можете також використовувати інші логічні зв'язки.

#### **Упорядкування рядків за допомогою arrange ()**

Arrange () працює аналогічно subset () за винятком того, що замість вибору рядків вона переупорядковує їх. Функція отримує на вхід назву сукупності даних і список імен колонок (або більш складний вираз) для упорядкування. Якщо буде вказано більше однієї колонки, то кожна наступна колонка буде упорядковуватися в межах кожного окремої сукупності значень попередніх.

#### **Вибір колонок за допомогою select ()**

Часто трапляється, що під час роботи з великою сукупністю даних насправді цікаві тільки деякі колонки. Select () дозволяє швидко сфокусуватися на частині колонок, використовуючи при цьому символічні назви колонок.

#### **Створення нових колонок за допомогою mutate ()**

Буває корисно не тільки вибрати окремі колонки, але й обчислити на їх підставі значення деякої іншої колонки. Для цих цілей призначена mutate () .

### **Обчислення підсумків за допомогою summarise ()**

Summarise () згортав дані сукупності в одну колонку.

Всі перераховані функції мають загальне:

- перший аргумент – data.frame;
- наступні аргументи описують те, що ви хочете зробити з першим, і ви можете звертатися до атрибутів першого аргументу за назвами, не використовуючи для доступу \$;
- повертається результат – новий data.frame.

Всі ці властивості разом дозволяють легко комбінувати безліч викликів в один ланцюжок для обчислення складного результату.

Ці п'ять функцій створюють основу для мови маніпулювання даними. На найпростішому рівні ви можете тільки змінити що-небудь в сукупності даних п'ятьма способами: змінити порядок рядків (arrange()), вибрati спостереження або атрибути (filter() i select()), додати обчислені атрибути (mutate()) або згорнути сукупність до підсумку (summarise()).

### **Операції угруповання**

Вищеописані функції корисні самі по собі, але особливої сили вони набувають в комбінації з концепцією угруповання, повторення обчислень дляожної групи спостережень окремо. В dplyr використовується функція group\_by() для розбиття сукупності даних на частини за рядками.

Можливо використовувати висновок функції group\_by() безпосередньо як перший аргумент перерахованих вище п'яти основних функцій, вони самі оброблять правильно розбитий на групи набір даних.

Dplyr пропонує ще кілька корисних функцій:

- N(): кількість спостережень в групі;
- N\_distinct(x): кількість спостережень з унікальним значенням x;
- First(x), last(x) i nth(x, n) – працюють подібно x [1], x [length(x)], i x [n] , але дають більше контролю над результатом якщо значення неможливо отримати.

## Data tables

Dplyr також пропонує всі перераховані методи маніпулювання даними та для об'єктів типу `data.tables`, ви просто повинні замінити набір даних на `data.table`.

`Data.table` може виявитися швидше в багатьох випадках, тому що можливе виконання декількох операцій одночасно. Наприклад, ви можете виконати `mutate` і `select` за один виклик, і `data.table` зрозуміє, що немає потреби розраховувати нову змінну в рядках, що повинні бути відфільтровані.

Переваги використання `data.tables` такі:

- у більшості випадків `data.tables` ізоляє вас безпосередньо від даних, і захищає їх у такий спосіб від ненавмисного зміни;
- замість використання вбудованого оператора пропонується багато відносно простих методів.

## Бази даних

Dplyr дозволяє використовувати віддалені бази даних так само як `data.frame`.

Порівняно з усіма існуючими альтернативами, dplyr:

- абстрагує від способу зберігання даних, так що можливе використання одного і того ж набору функцій для маніпулювання `data.frame`, `data.tables` і віддаленими базами даних;
- пропонує продуманий метод `print()`, що не роздрукує випадково кілька сторінок даних на екран.

Порівняно з функціями за замовченням:

- dplyr набагато більш стрункий; функції мають ідентичний інтерфейс;
- функції за замовченням прагнуть обробляти вектори; dplyr концентрується на сукупностях даних.

## Додаток 4. Пакет `ggplot2`

### ggplot2

#### Download the data

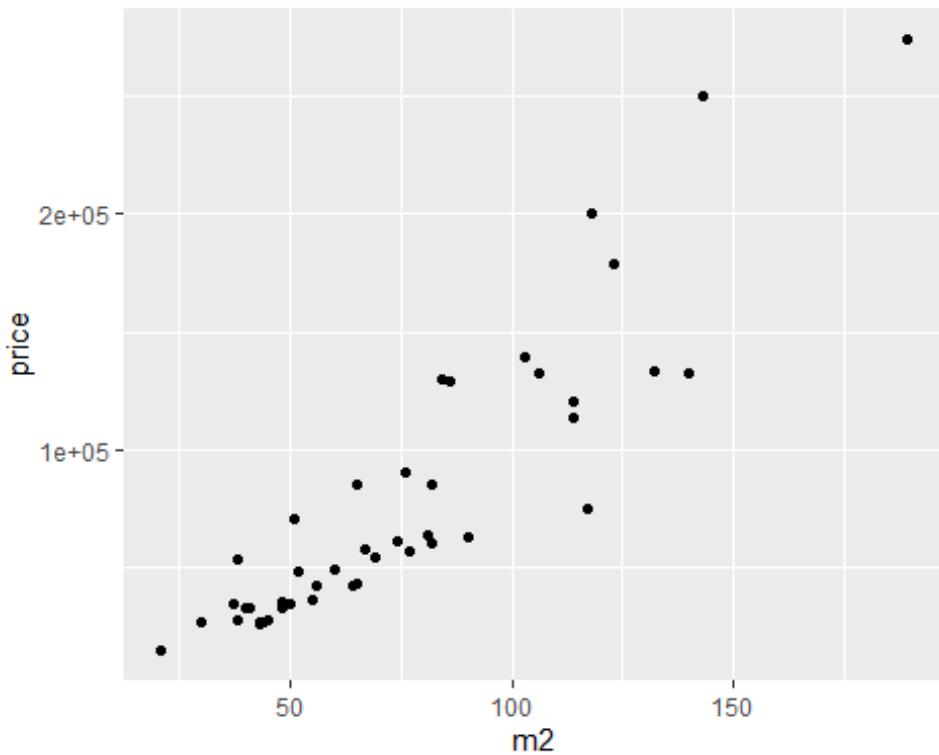
```
#Source files are here
#setwd('D:/ML')
#Download the files
f1 <- read.csv2('flats_test.csv', header = TRUE, encoding = 'UNICOD')
f2 <- read.csv2('flats_fit.csv', header = TRUE, encoding = 'UNICOD')
f2 <- f2[-1]
f <- dplyr::bind_cols(f1, f2)
f <- f[,-1]
f$type <- as.factor(f$type)
head(f)

##   rooms location    condition m2 type price      p_sr      p_mr      p_
pr
## 1     1   center     repaired 37     2 35000 23858.08 44860.80 31967.28
## 2     2 suburbs     repaired 21     2 15000 1721.26 3504.32 22337.07
## 3     1 suburbs     repaired 82     1 60000 86117.84 102194.70 80639.08
## 4     3   center     repaired 82     2 85000 86117.84 95070.28 80639.08
## 5     1   center unrepaired 41     1 33000 29392.29 32799.70 35090.37
## 6     1   center unrepaired 40     1 33000 28008.78 31317.12 34284.27
##       p_dt      p_rf
## 1 35960.83 32465.71
## 2 35960.83 32275.88
## 3 83140.39 72974.14
## 4 83140.39 72974.14
## 5 35960.83 31591.74
## 6 35960.83 32181.48
```

#### Basic tasks

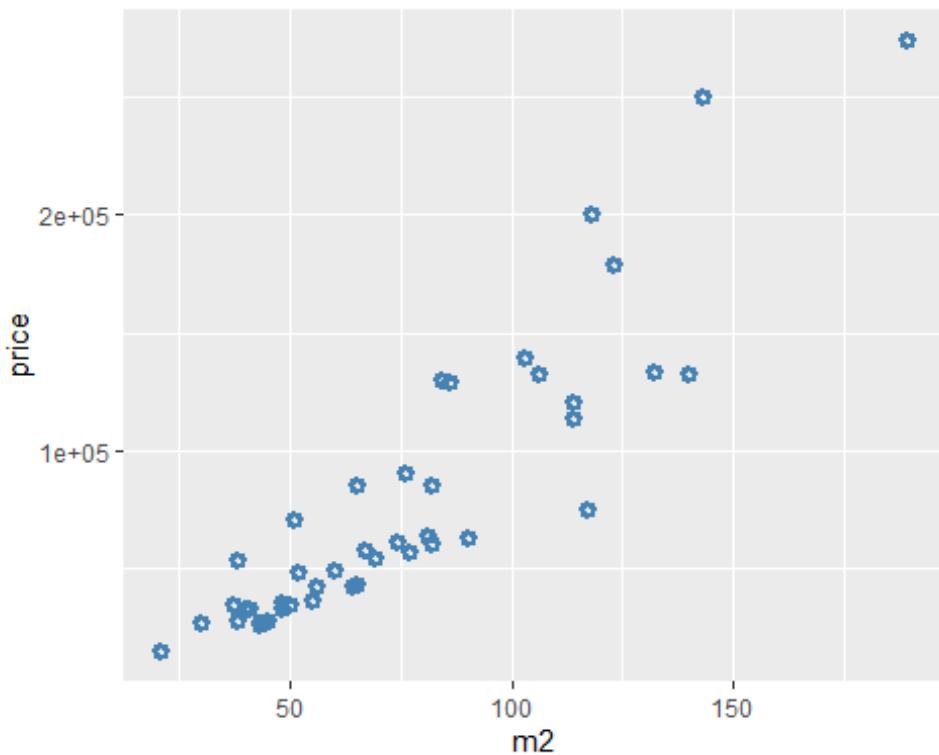
##### Basic plot setup

```
library(ggplot2)
gg <- ggplot(f, aes(x=m2, y=price))
gg + geom_point()
```



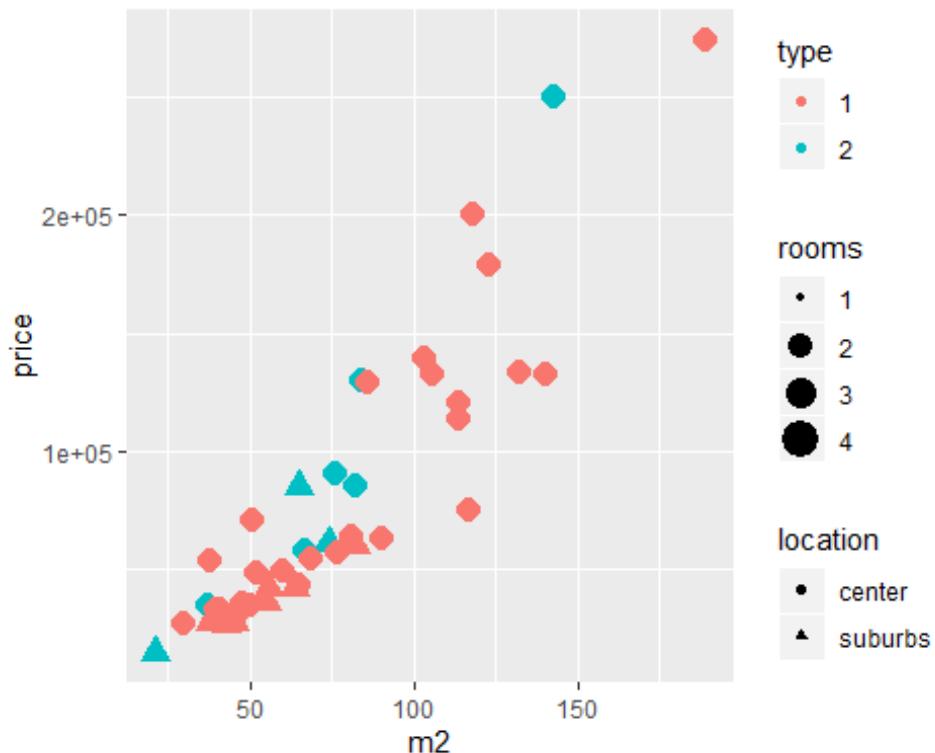
### Static – point size, shape, color and boundary thickness

```
gg + geom_point(size=1, shape=1, color="steelblue", stroke=2) # 'stroke'  
controls the thickness of point boundary
```



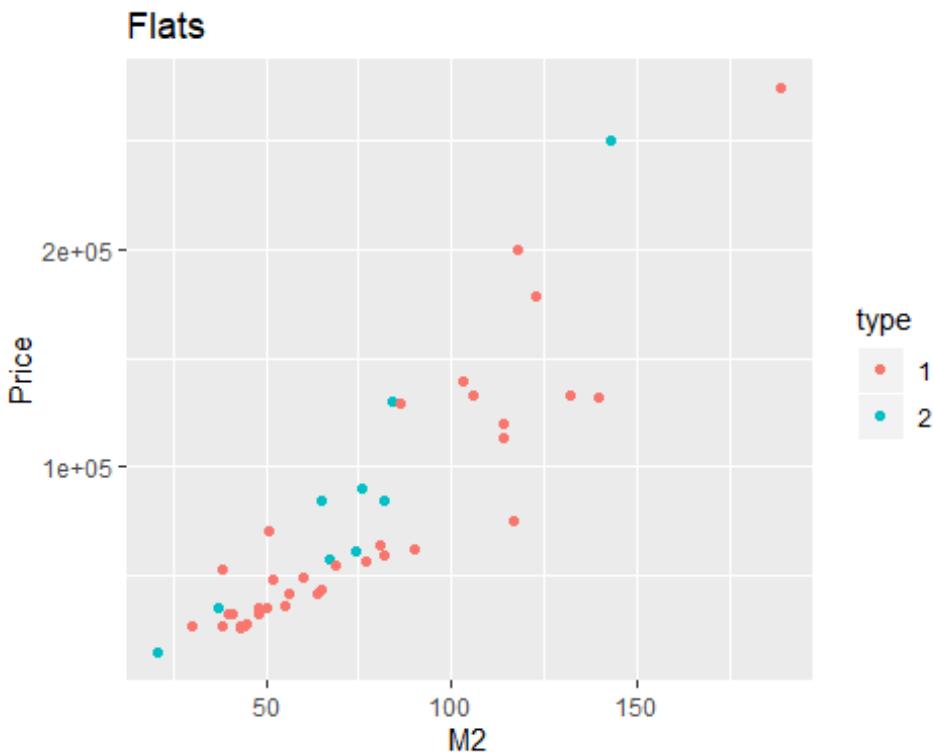
## Dynamic – point size, shape, color and boundary thickness

```
#Make the aesthetics vary based on a variable.  
gg + geom_point(aes(size=rooms, shape=location, color=type, stroke=condition))  
  
## Warning in Ops.factor(coords$stroke, .stroke): '*' not meaningful for  
## factors  
  
## Warning in Ops.factor(coords$stroke, .stroke): '*' not meaningful for  
## factors
```



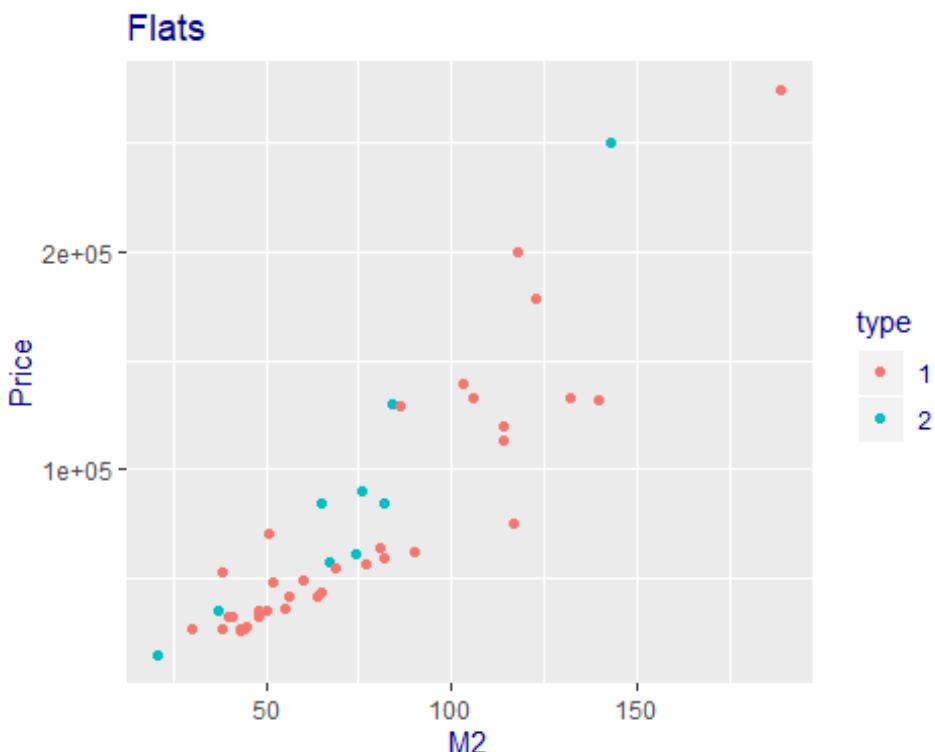
## Add Title, X and Y axis labels

```
gg1 <- gg + geom_point(aes(color=type))  
gg2 <- gg1 + labs(title="Flats", x="M2", y="Price")  
print(gg2)
```



### Change color of all text

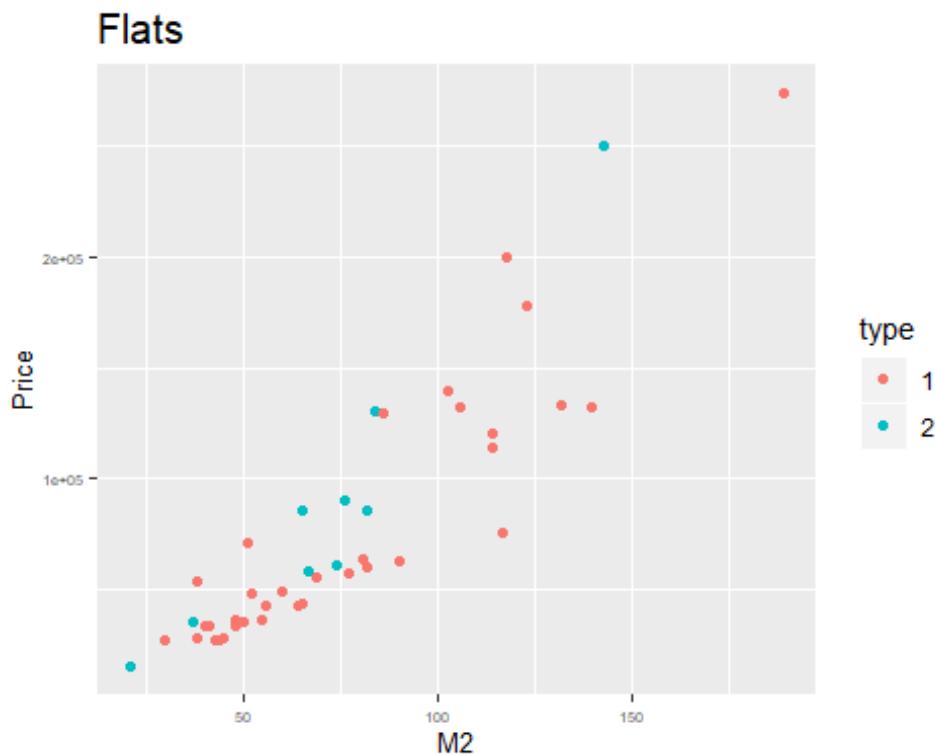
```
gg2 + theme(text=element_text(color="dark blue")) # all text turns blue.
```



### Change title, X and Y axis label and text size

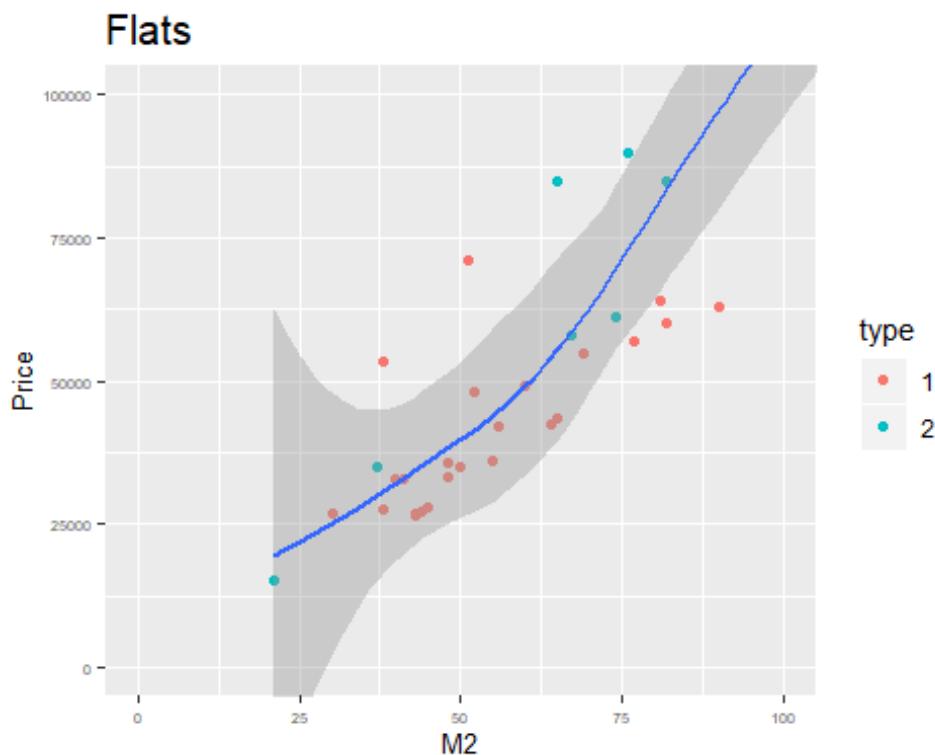
```
gg3 <- gg2 + theme(plot.title=element_text(size=15), axis.title.x=element_
text(size=10), axis.title.y=element_text(size=10), axis.text.x=element_tex
```

```
t(size=5), axis.text.y=element_text(size=5))  
print(gg3)
```



### Adjust X and Y axis limits

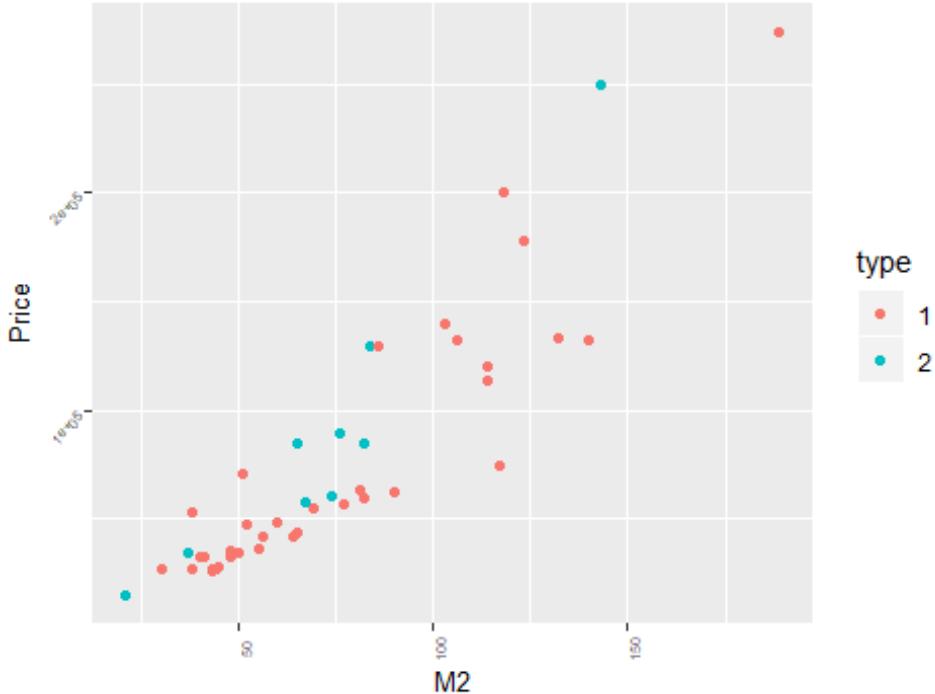
```
gg3 + coord_cartesian(xlim=c(0,100), ylim=c(0, 100000)) + geom_smooth()  
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



### Rotate axis text

```
gg3 + theme(axis.text.x=element_text(angle=90), axis.text.y=element_text(a  
ngle=45))
```

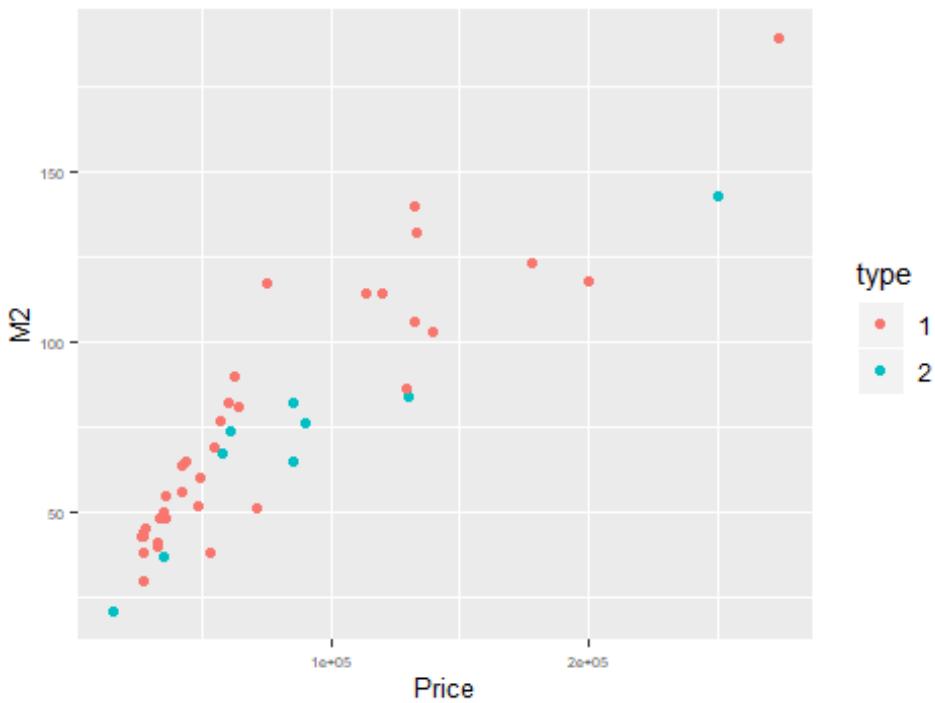
Flats



### Flip X and Y Axis

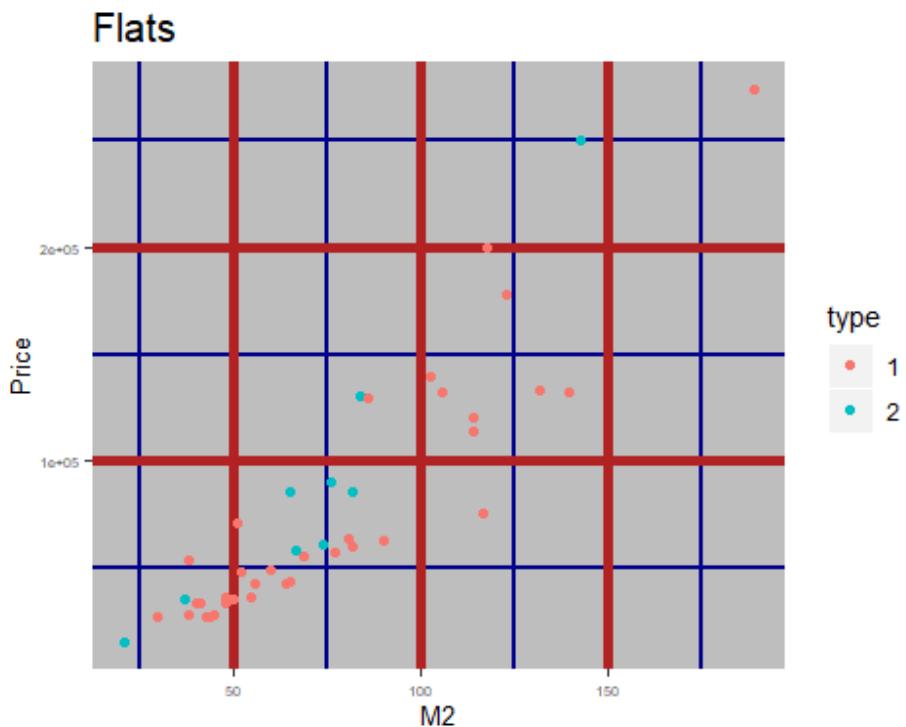
```
gg3 + coord_flip() # flips X and Y axis.
```

Flats



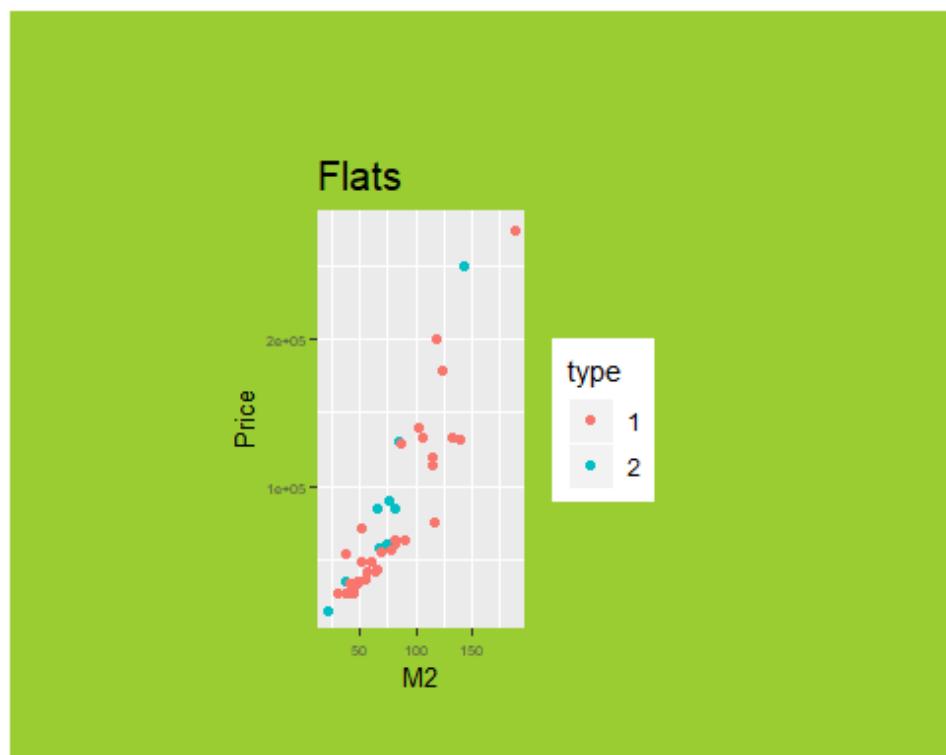
### Grid lines and panel background

```
gg3 + theme(panel.background = element_rect(fill = 'gray'),
            panel.grid.major = element_line(colour = "firebrick", size=2),
            panel.grid.minor = element_line(colour = "dark blue", size=1))
```



### Plot margin and background

```
gg3 + theme(plot.background=element_rect(fill="yellowgreen"),
            plot.margin = unit(c(2, 4, 1, 3), "cm")) # top, right, bottom, left
```



## #Colors

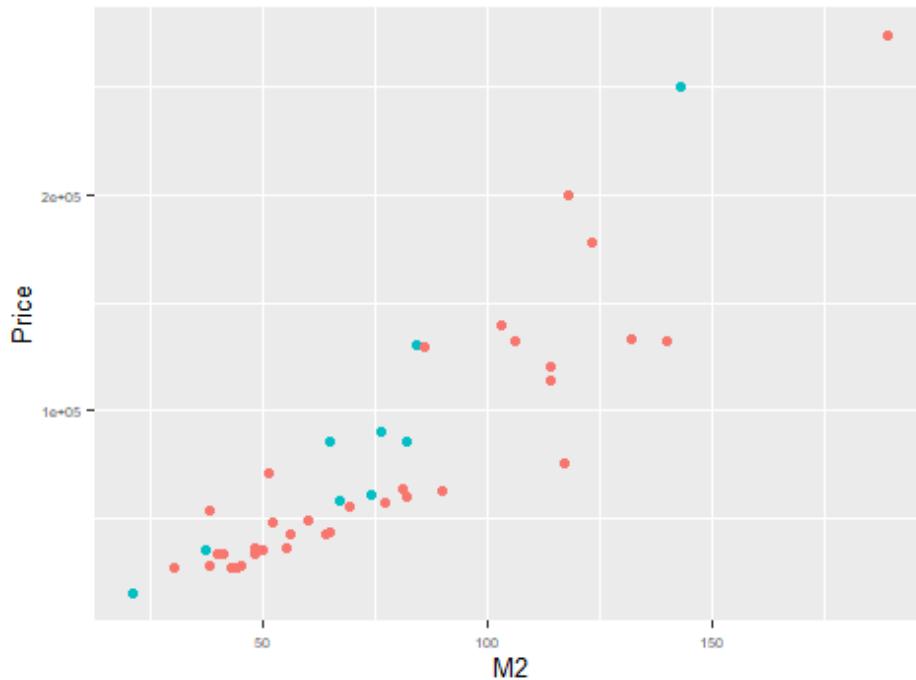
*#The whole list of colors are displayed at your R console in the color() function. Here are few of my suggestions for nice looking colors and backgrounds:*

```
# steelblue (points and lines)
# firebrick (point and lines)
# springgreen (fills)
# violetred (fills)
# tomato (fills)
# skyblue (bg)
# sienna (points, lines)
# slateblue (fills)
# seagreen (points, lines, fills)
# sandybrown (fills)
# salmon (fills)
# saddlebrown (lines)
# royalblue (fills)
# orangered (point, lines, fills)
# olivedrab (points, lines, fills)
# midnightblue (lines)
# mediumvioletred (points, lines, fills)
# maroon (points, lines, fills)
# Limegreen (fills)
# Lawngreen (fills)
# forestgreen (lines, fills)
# dodgerblue (fills, bg)
# dimgray (grids, secondary bg)
# deeppink (fills)
# darkred (lines, points)
```

**Legend****Hide legend**

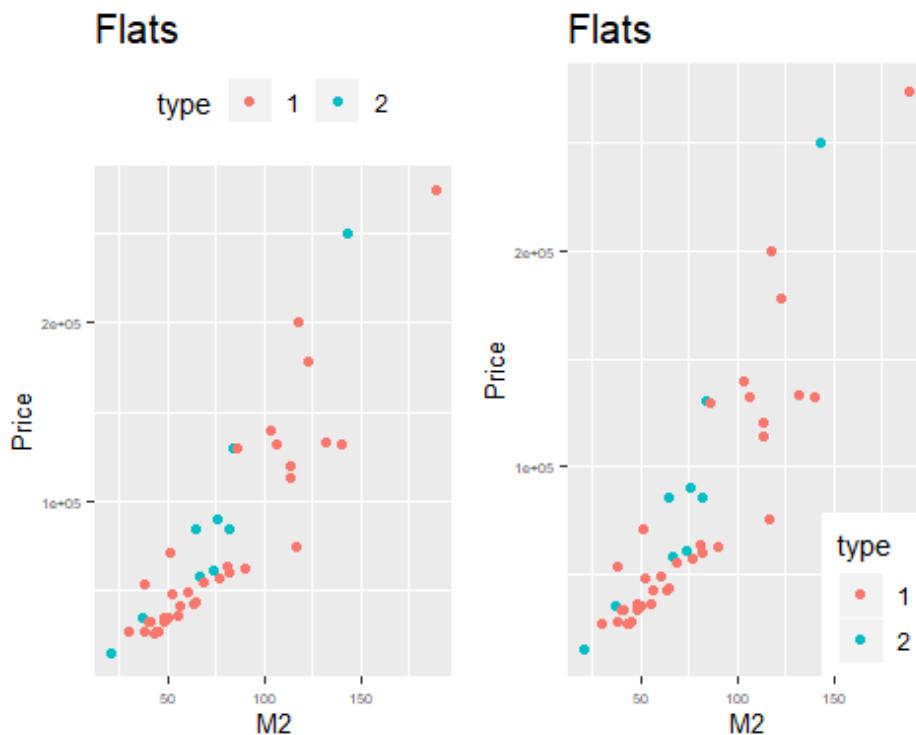
```
gg3 + theme(legend.position="none") # hides the legend
```

## Flats



## Change legend position

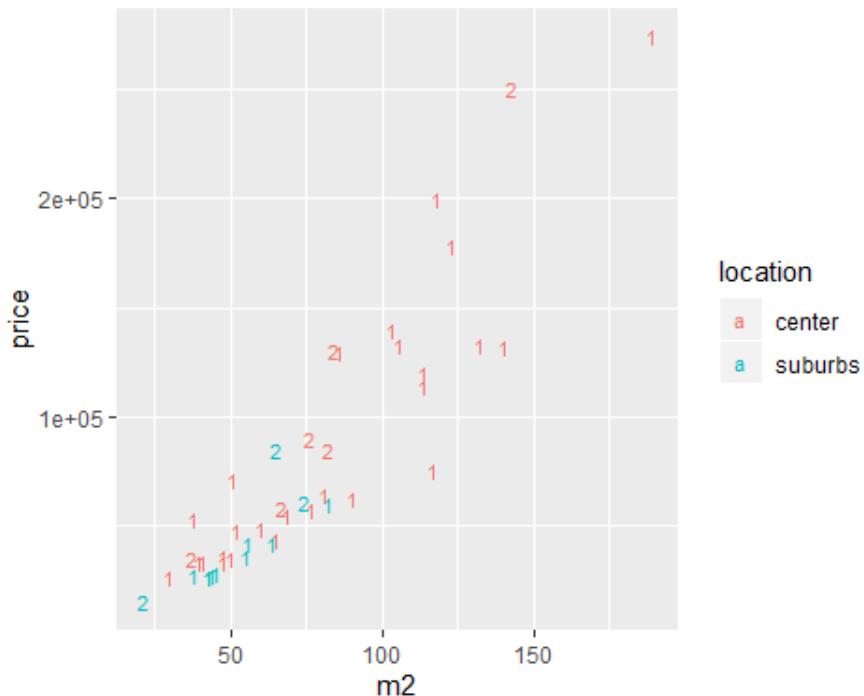
```
#Outside plot
p1 <- gg3 + theme(legend.position="top") # top / bottom / left / right
#Inside plot
p2 <- gg3 + theme(legend.justification=c(1,0), legend.position=c(1,0)) # Legend justification is the anchor point on the legend, considering the bottom Left of Legend as (0,0)
gridExtra::grid.arrange(p1, p2, ncol=2)
```



## Plot text and annotation

### Add text in chart

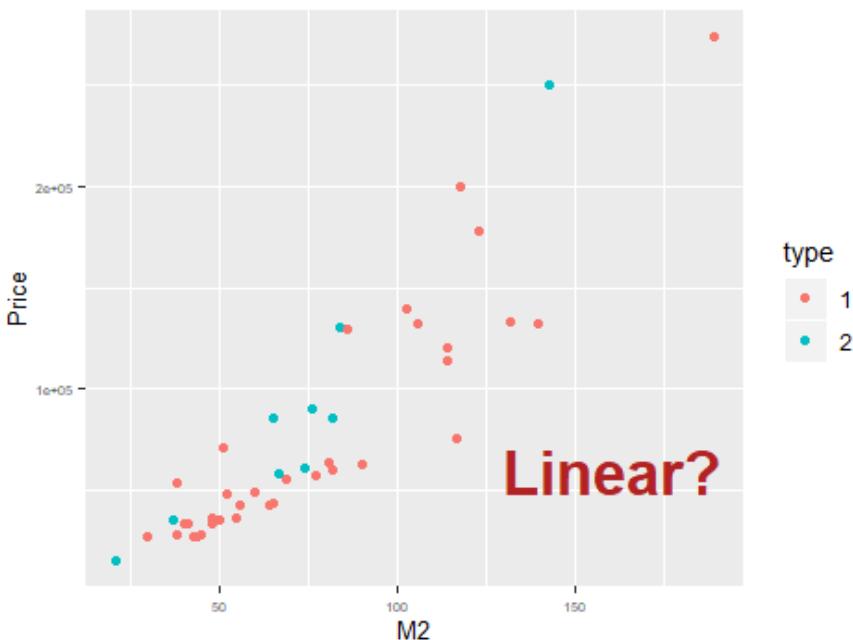
```
gg + geom_text(aes(label=type, color=location), size=3)
```



### Annotation

```
library(grid)
my_grob = grobTree(textGrob("Linear?", x=0.8, y=0.2, gp=gpar(col="firebrick",
k", fontsize=25, fontface="bold")))
gg3 + annotation_custom(my_grob)
```

### Flats

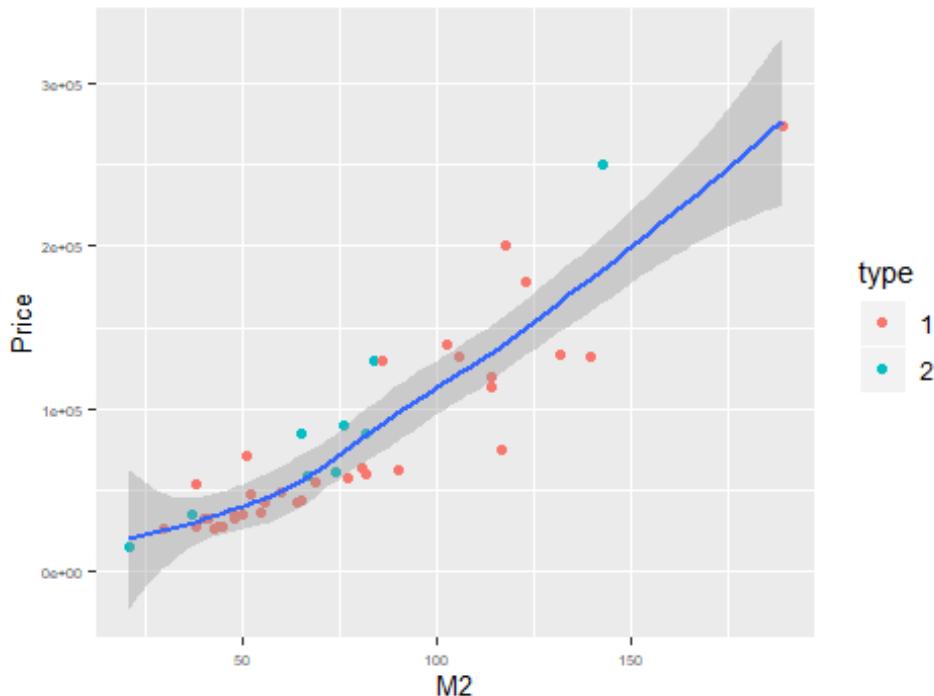


## Geom layers

### Add smoothing line

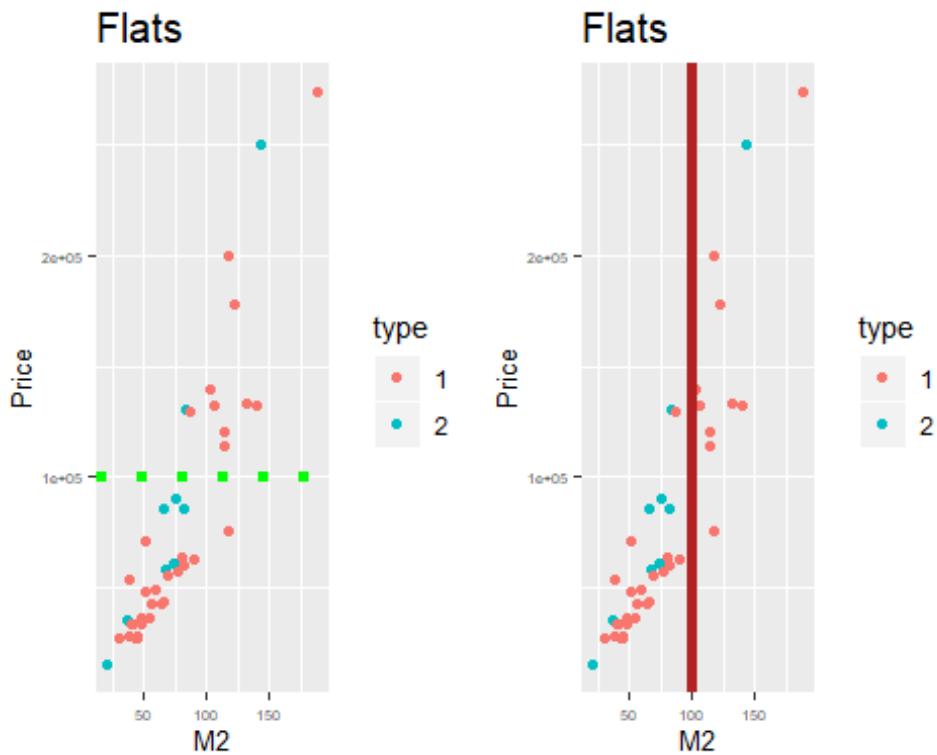
```
gg3 + geom_smooth(aes()) # method could be - 'Lm', 'Loess', 'gam'  
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

Flats



### Add horizontal/vertical line

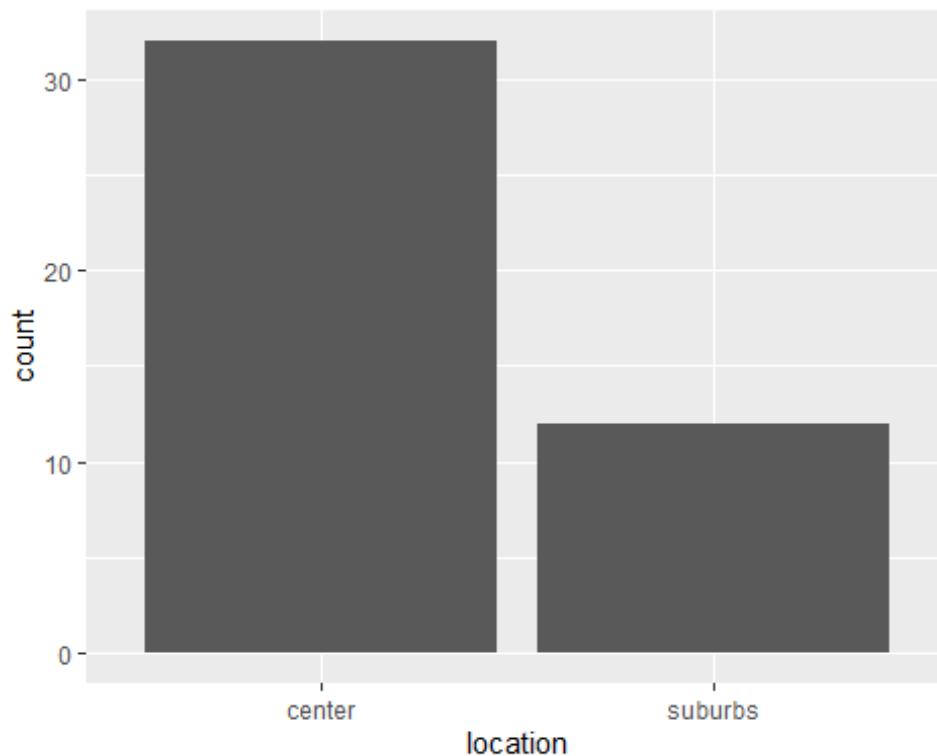
```
p1 <- gg3 + geom_hline(yintercept=100000, size=2, linetype="dotted", color="green") # linetypes: solid, dashed, dotted, dotdash, Longdash and twodash  
p2 <- gg3 + geom_vline(xintercept=100, size=2, color="firebrick")  
gridExtra::grid.arrange(p1,p2, ncol=2)
```



## Add bar chart

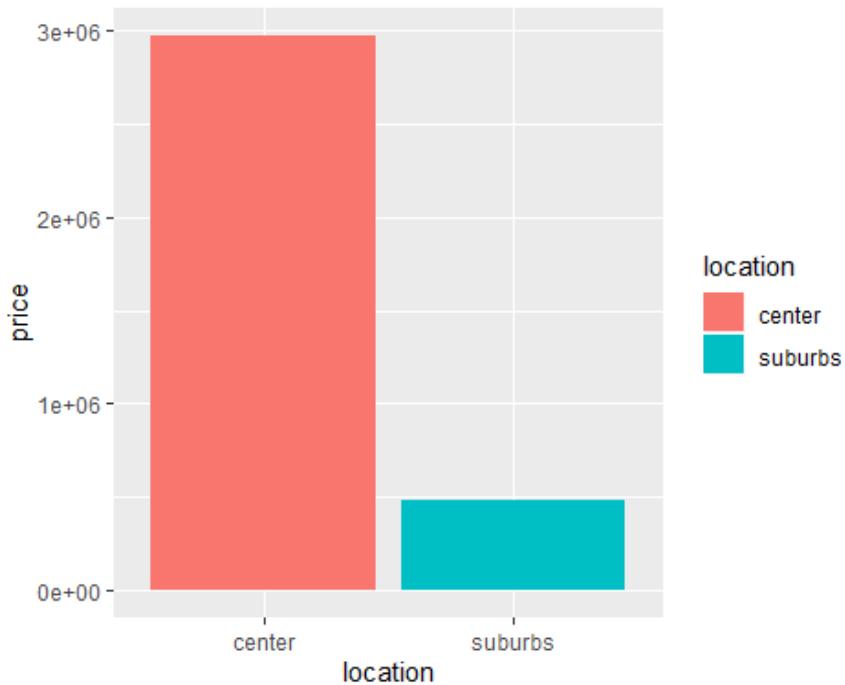
**Frequency bar chart: Specify only X axis.**

```
gg <- ggplot(f, aes(x=location))
gg + geom_bar() # frequency table
```



### Distinct color for bars

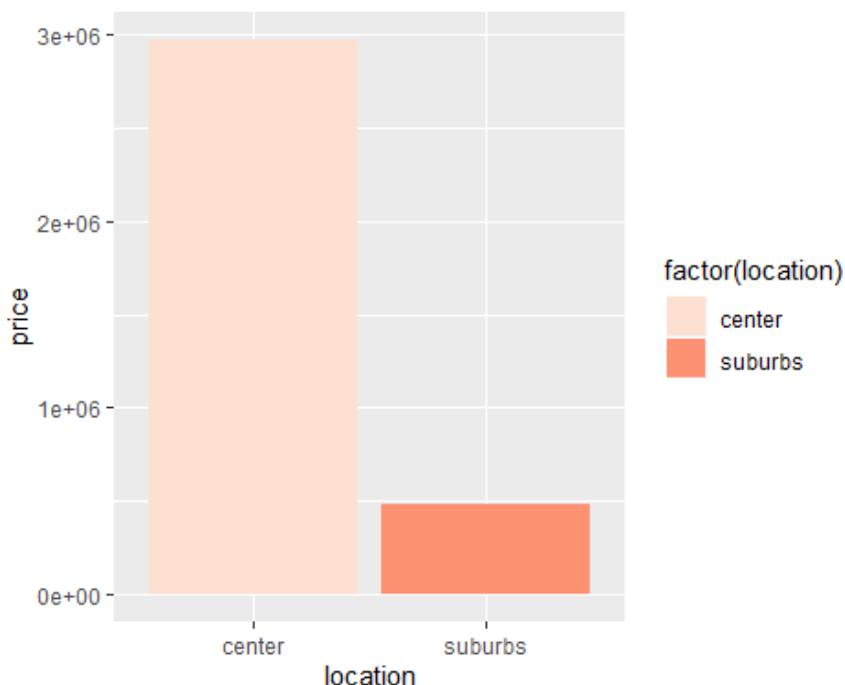
```
gg_bar <- ggplot(f, aes(x=location, y=price)) + geom_bar(stat = "identity", aes(fill=location))
print(gg_bar)
```



### Change color palette

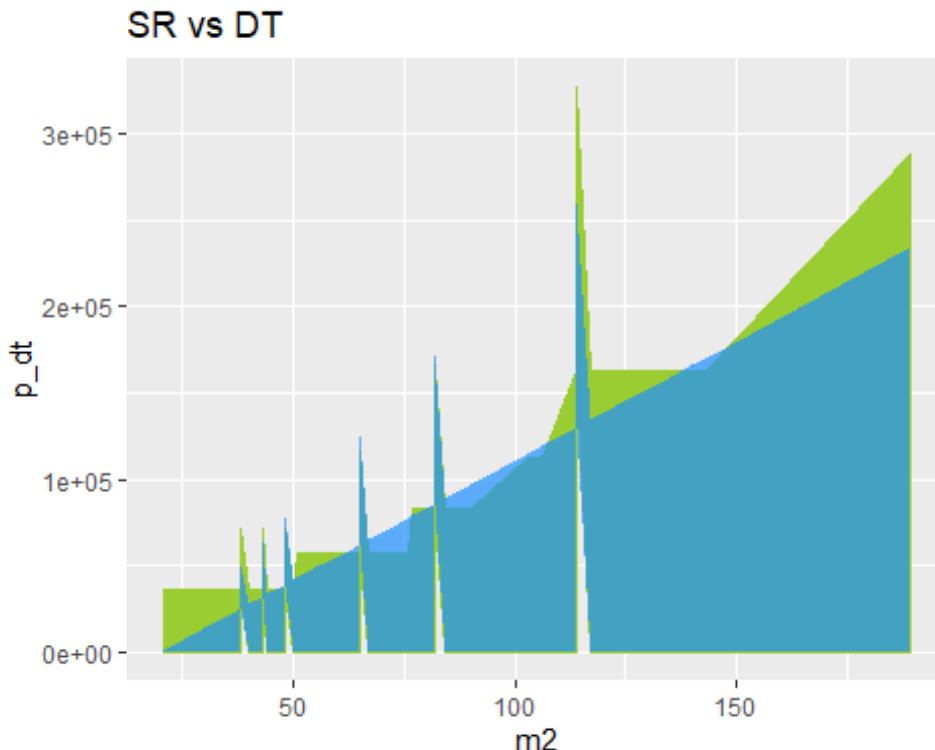
```
library(RColorBrewer)
display.brewer.all(n=20, exact.n=FALSE)

ggplot(f, aes(x=location, y=price, fill=factor(location))) + geom_bar(stat = "identity") + scale_fill_brewer(palette="Reds") # "Reds" is palette name
```



## Area

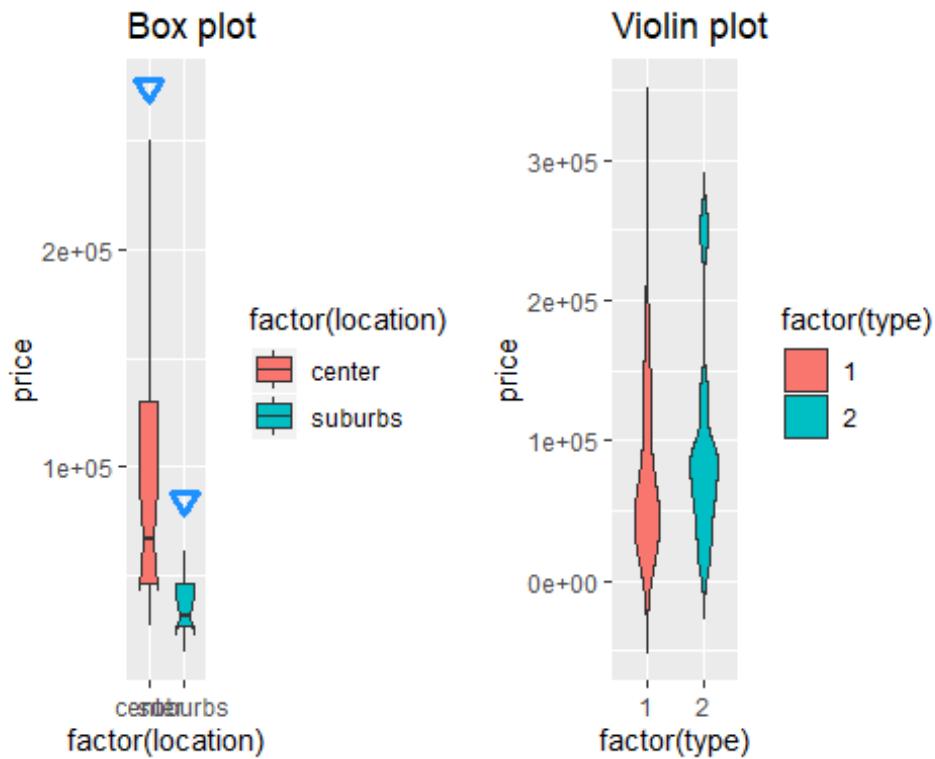
```
ggplot(f, aes(x=m2)) + geom_area(aes(y=p_dt), fill="yellowgreen", color="yellowgreen") + geom_area(aes(y=p_sr), fill="dodgerblue", alpha=0.7, linetype="dotted") + labs(title="SR vs DT")
```



## Boxplot and Violin

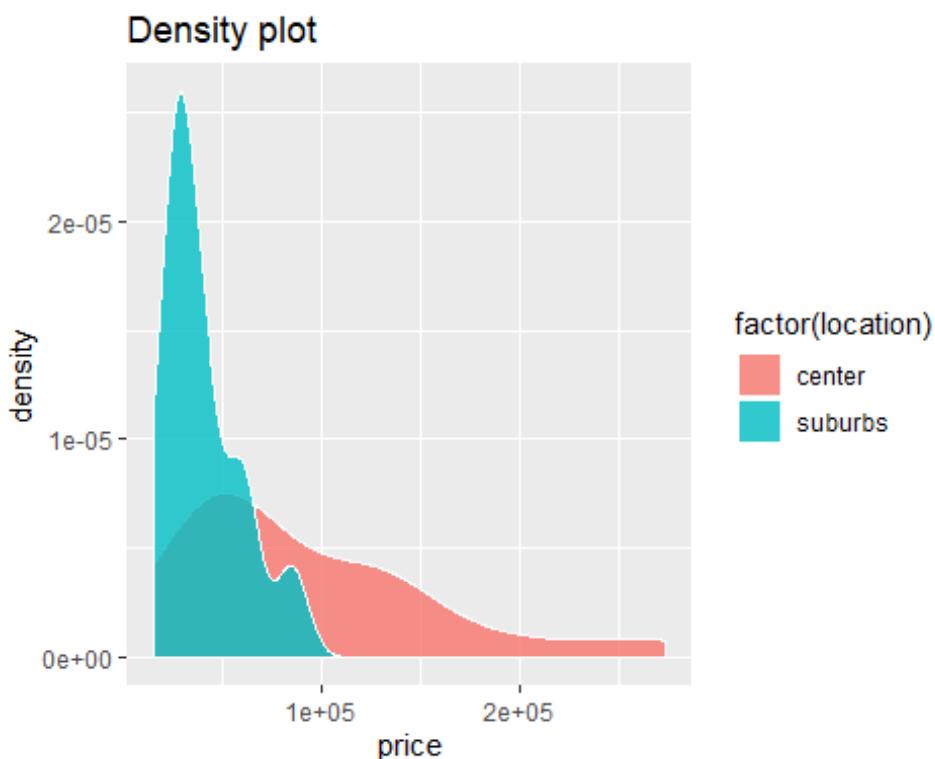
```
p1 <- ggplot(f, aes(factor(location), price)) + geom_boxplot(aes(fill = factor(location)), width=0.5, outlier.colour = "dodgerblue", outlier.size = 2, outlier.shape = 6, outlier.stroke = 2, notch=T) + labs(title="Box plot")
p2 <- ggplot(f, aes(factor(type), price)) + geom_violin(aes(fill = factor(type)), width=0.5, trim=F) + labs(title="Violin plot")
gridExtra::grid.arrange(p1, p2, ncol=2)

## notch went outside hinges. Try setting notch=FALSE.
## notch went outside hinges. Try setting notch=FALSE.
```



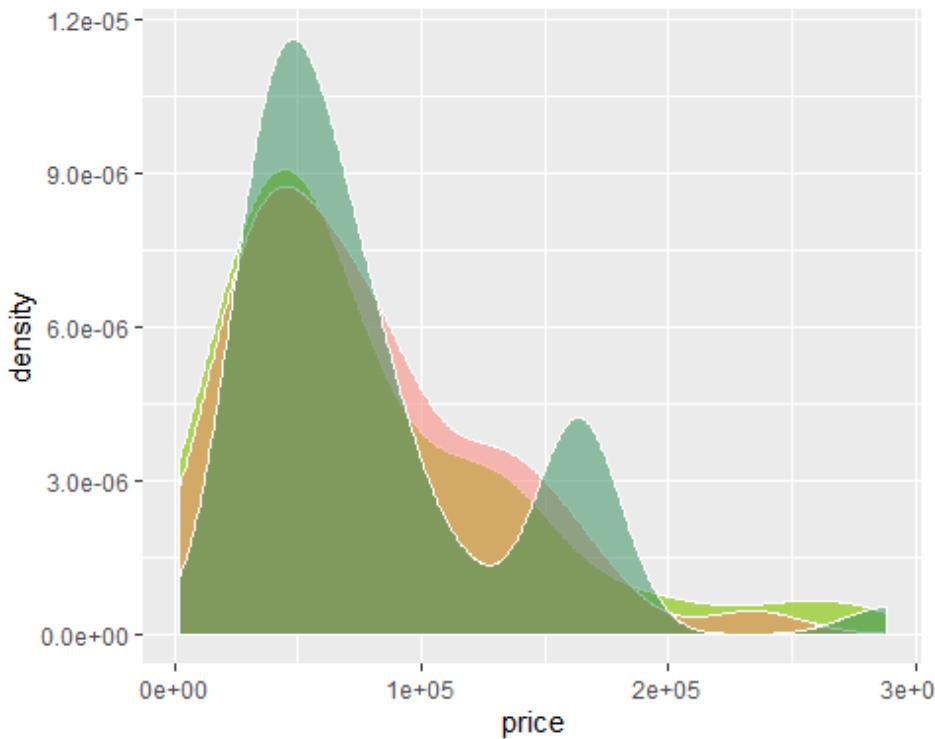
### Density

```
ggplot(f, aes(price)) + geom_density(aes(fill = factor(location)), color="white", alpha=0.8, size=0.5) + labs(title="Density plot")
```



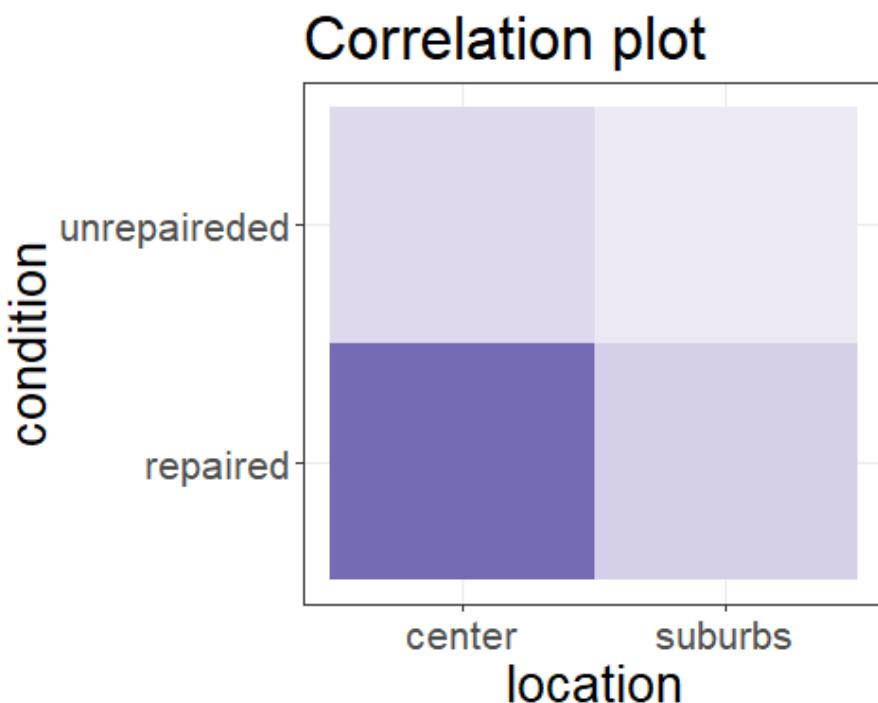
```
d1 <- ggplot(f) + geom_density(aes(x=price), fill="yellowgreen", color="white", alpha=0.8) + geom_density(aes(x=p_sr), fill="salmon",
```

```
color="white", alpha=0.5) + geom_density(aes(x=p_dt), fill="seagreen",
color="white", alpha=0.5)
```



## Tiles

```
library(RColorBrewer)
ggplot(f, aes(x=location, y=condition, fill=price, label=price)) + geom_tile() + theme_bw() + labs(title="Correlation plot") + theme(text=element_text(size=20), legend.position="none") + scale_fill_gradient2()
```



## Додаток 5. Пакет 'ROCR'

ROCR – пакет R для оцінки та візуалізації якості класифікації. Є можливість підрахунку різних мір якості класифікації та побудови 2D графіків для окремих змінних або для залежності однієї змінної від іншої.

Для використання засобів пакета необхідні дані про справжні (labels) і передбачені будь-яким чином (predictions) мітки класів вибірки об'єктів.

Водночас дані можуть являти собою опис як однієї вибірки, так і декількох, наприклад, кількох підвибірок із використанням крос-валідації. Сукупності справжніх і передбачених міток подаються у вигляді векторів або списків однакової довжини. У разі декількох вибірок дані подаються у вигляді матриці або фрейму, в якому кожен стовпець являє собою окрему вибірку.

Справжні мітки можуть набувати тільки двох значень, на яких має задаватися відношення порівняння. Якщо прогнозні та справжні мітки є числами, то прогнозні мітки можуть набувати скільки завгодно різних значень, якщо прогнозні або справжні мітки не є числами, то прогнозні мітки можуть набувати тільки 2 значення. Клас, для якого мітка більше, будемо називати позитивним, а інший – негативним.

### Міри якості

acc	Accuracy
err	Error rate
fpr, fall	False positive rate, fallout
tpr, rec, sens	True positive rate, recall, sensitivity.
fnr, miss	False negative rate, miss.
tnr, spec	True negative rate, specificity.
ppv, prec	Positive predictive value, precision.
npv	Negative predictive value
pcfall	Prediction-conditioned fallout
pcmiss	Prediction-conditioned miss
rpp	Rate of positive predictions
rnp	Rate of negative predictions
phi, mat	Phi correlation coefficient, matthews correlation coefficient. Дає число від 1 до 1, де 1 – ідеальний прогноз, 0 вказує на випадковий прогноз. Величини нижче за 0 вказують на прогноз гірше за випадковий

mi	Mutual information
chisq	Chi square test statistic
odds	Odds ratio
lift	Lift value
f	Precision-recall F measure. Виражається через precision (P), recall (R) і деякий задається коефіцієнт $\alpha$ з відрізка [0,1]. Значення $\alpha$ за замовчуванням 0.5
rch	ROC convex hull. ROC криві (tpr vs fpr) з віддаленими увігнутими кривими (отриманими у разі неоптимальних значень відсікання). Не може використовуватися спільно з іншими мірами
auc	Area under the ROC curve. Не може використовуватися спільно з іншими мірами. Можливий підрахунок часткової площини до певного значення fpr, для цього потрібно подавати параметр fpr.stop з відрізка [0,1]
prbe	Precision-recall break-even point. Відсічення, в яких precision і recall рівні. Не може використовуватися спільно з іншими мірами
cal	Calibration error. Абсолютна різниця між точністю передбачення міток і результатом класифікації. Ця міра обчислюється для всіх міток шляхом проходу вікном по всьому простору відміток. Розмір вікна за замовчуванням 100, його можна задати за допомогою додаткового параметра window.size. Використовується тільки з ймовірними прогнозами (від 0 до 1)
mxe	Mean cross-entropy. Використовується тільки з ймовірними прогнозами (від 0 до 1). Не може використовуватися спільно з іншими мірами
rmse	Root-mean-squared error. Використовується тільки з числовими позначками класів. Не може використовуватися спільно з іншими мірами
sar	Комбінація декількох мір для отримання більш стійкої міри. 1/3 * (Accuracy + Area under the ROC curve + Root mean-squared error)
ecost	Expected cost. Має обов'язкову вісь $x$ – probability-cost function. Не може використовуватися спільно з іншими мірами
cost	Вартість класифікатора при явно заданих цінах за помилку на позитивному і негативному класах. Доступні додаткові параметри cost.fp і cost.fn, які і задають ціну помилкової класифікації в позитивний і негативний клас відповідно. За замовчуванням обидва параметри рівні 1.

де  $P/N$  – кількість об'єктів, що належать до позитивного/негативного класу;

$TP/TN$  – кількість об'єктів, правильно долучених до позитивного/негативного класу;

$FP/FN$  – кількість об'єктів, помилково долучених до позитивного/негативного класу.

Часто використовувані разом пари мір:

- ROC curves: tpr vs fpr;
- Precision/recall graphs: prec vs rec;
- Sensitivity/specificity plots: sens vs spec;
- Lift charts: lift vs gpp.

Пакет містить два класи: prediction та performance.

#### *prediction class*

Об'єкти цього класу призначені для внутрішнього подання вихідних даних: істинних і передбачених якимось чином міток класів.

Компоненти:

Всі списки можуть мати кілька елементів, якщо вихідні дані складаються з більш, ніж однієї вибірки.

<code>predictions</code>	Список, кожен елемент якого являє собою вектор передбачених міток
<code>labels</code>	Список, кожен елемент якого являє собою вектор дійсних міток
<code>cutoffs</code>	Список, кожен елемент якого являє собою вектор всіх можливих передбачених міток (водночас додається значення Inf, мітки сортуються в порядку убування і видаляються повтори)
<code>fp</code>	Список, кожен елемент якого являє собою вектор, який складається з кількості неправильно класифікованих об'єктів позитивного класу під час поділу об'єктів на основі передбачених міток на класи за <code>cutoffs</code>
<code>tp</code>	Те ж, що <code>fp</code> , але для правильно класифікованих об'єктів позитивного класу
<code>tn</code>	Те ж, що <code>fp</code> , але для правильно класифікованих об'єктів негативного класу
<code>fn</code>	Те ж, що <code>fp</code> , але для неправильно класифікованих об'єктів негативного класу
<code>n.pos</code>	Список, кожен елемент якого містить кількість об'єктів позитивного класу за істиними мітками
<code>n.neg</code>	Те ж, що <code>n.pos</code> , але для об'єктів негативного класу
<code>n.pos.pred</code>	Список, кожен елемент якого являє собою вектор, який складається з кількості об'єктів, що належать до позитивного класу у разі поділу об'єктів на основі передбачених міток на класи за <code>cutoffs</code>
<code>n.neg.pred</code>	Те ж, що <code>n.pos.pred</code> , але для негативного класу

*performance class*

Об'єкти цього класу призначені для зберігання результатів оцінки якості класифікації в формі призначеної для побудови графіка (окремо розглядаються заходи якості для осей і параметризація).

Компоненти:

Всі списки можуть мати кілька елементів, якщо вихідні дані складаються з більш, ніж однієї вибірки.

x.name	Назва міри якості, використовуваної для осі x
y.name	Назва міри якості, використовуваної для осі y
alpha.name	Назва елемента, використовуваного для створення параметризованої кривої. Зазвичай це "none" або "cutoff"
x.values	Список, кожен елемент якого являє собою вектор, що складається зі значень міри якості x в точках відповідного вектора alpha.values
y.values	Те ж, що x.values, але для міри якості y
alpha.values	Список, кожен елемент якого являє собою вектор, що складається зі значень заданого параметра кривої

Об'єкт цього класу може мати 4 різні види (для кожного виду наведено приклад його створення за допомогою функції `performance`, опис якої буде нижче):

- описується поведінка міри, що залежить від відсічення, на всіх можливих значеннях відсічення. Тоді в x записуються значення відсічення, а в y – значення міри, alpha залишається порожньою. Приклад: `performance(pred, "acc")`;
- описується залежність двох мір, що залежать від відсічення. Тоді в x та y записуються значення заходів, а в alpha – відсічення. Приклад: `performance(pred, "tpr", "fpr")`;
- описується поведінка міри, для якої спочатку закладена деяка друга вісь. Тоді в x записуються значення за цією другою віссю, а в y – міри, alpha залишається порожньою. Приклад: `performance(pred, "ecost")`;

- описується поведінка міри, що є скаляром. Тоді в  $y$  записуються значення міри, а  $x$  та  $\alpha$  залишаються порожніми. Приклад: `performance(pred, "auc")`.

Пакет містить три функції: `prediction`, `performance` і `plot`.

### *prediction*

Функція для створення об'єкта класу `prediction` з вихідних даних.

Виклик:

`prediction(predictions, labels, label.ordering = NULL)`

Аргументи:

<code>predictions</code>	Вектор, матриця, список або фрейм, що містить передбачені мітки вибірки об'єктів.
<code>labels</code>	Вектор, матриця, список або фрейм, що містить справжні мітки вибірки об'єктів.
<code>label.ordering</code>	Відношення порівняння між мітками класу за замовчуванням можна змінити, вважаючи за аргумент вектор, що містить мітки негативного та позитивного класу.

### *performance*

Функція для створення об'єкта класу `performance` з об'єкта класу `prediction`.

Виклик: `performance(prediction.obj, measure, x.measure = "cutoff", ...)`

Аргументи:

<code>prediction.obj</code>	Об'єкт класу <code>prediction</code>
<code>measure</code>	Міра якості, що використовується для осі $y$
<code>x.measure</code>	Міра якості, що використовується для осі $x$
...	Додаткові аргументи, що визначені для деяких мір

### *plot*

Функція для візуалізації об'єкта класу `performance`.

Виклик: `plot(x, y, ..., avg = "none", spread.estimate = "none", spread.scale = 1, show.spread.at = c(), colorize = FALSE, colorize.palette = rev(rainbow(256, start = 0, end = 4/6)), colorkey = colorize, colorkey.relwidth =`

*0.25, colorkey.pos = "right", print.cutoffs.at = c (), cutoff.label.function = function (x) {round (x, 2)}, downsampling = 0, add = FALSE)*

Аргументи:

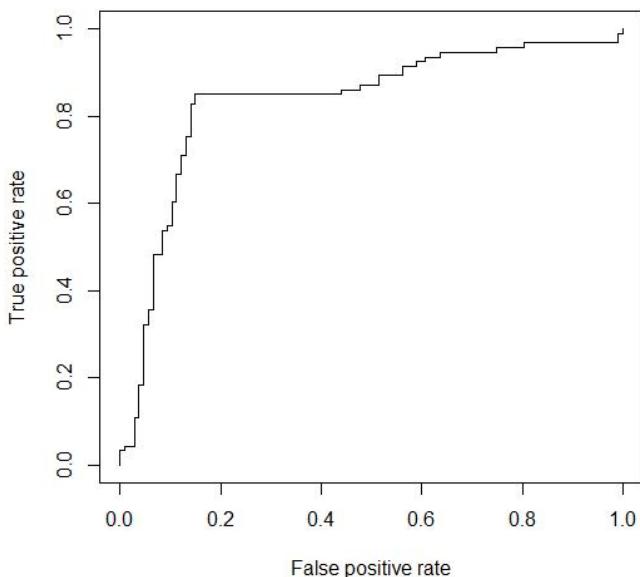
x	Об'єкт класу <code>performance</code> .
y	Не використовується
...	Додаткові графічні параметри для налаштування різних компонент графіка. Для звернення до параметру деякої компоненти потрібно користуватися таким записом: <code>component.parameter</code> . Доступні наступні компоненти: <code>xaxis</code> , <code>yaxis</code> , <code>coloraxis</code> , <code>box</code> , <code>points</code> , <code>text</code> , <code>plotCI</code> (похибки), <code>boxplot</code> . Під час налаштування параметрів самого полотна і кривих префікс вказувати не потрібно
avg	Об'єкт, що описує відображення декількох кривих (наприклад, якщо дані містять декілька вибірок об'єктів, отриманих під час крос-валідації, то і кривих виходить кілька). Криві можна усереднювати різними способами: <ul style="list-style-type: none"> <li>• <code>none</code> – криві малюються окремо без усереднення;</li> <li>• <code>horizontal</code> – горизонтальне усереднення;</li> <li>• <code>vertical</code> – вертикальне усереднення;</li> <li>• <code>threshold</code> – усереднення за відсіченням.</li> </ul>
spread.estimate	У разі усереднення кривих відхилення від середньої кривої може бути візуалізовано як: <ul style="list-style-type: none"> <li>• <code>stderror</code> – вікно стандартної помилки;</li> <li>• <code>stddev</code> – вікно стандартного відхилення;</li> <li>• <code>boxplot</code> – вікно розкиду.</li> </ul>
spread.scale	Константа, на яку додаються довжини вікон <code>stderror</code> і <code>stddev</code> .
show.spread.at	У разі вертикального усереднення цей вектор задає позиції <code>x</code> , в яких проводиться візуалізація. За замовчанням вона розраховується в 11 рівномірно розподілених по всьому простору значень <code>x</code> точках.
colorize	Логічне значення, що показує, чи повинна крива бути розфарбована відповідно до відсічення.
colorize.palette	Якщо <code>colorize</code> увімкнено, то визначає колірну палітру, в якій відображається діапазон відміток.
colorkey	Логічне значення. Якщо <code>TRUE</code> , то в 4-відсотковому граничному діапазоні малюється кольоровий ключ, що показує відображення відсічок в колірну палітру.
colorkey.relwidth	Константа від 0 до 1, що визначає частину 4-відсоткової граничної зони, що відводиться під кольоровий ключ.
colorkey.pos	Визначає, як малюється кольоровий ключ: вертикально справо або горизонтально зверху.
print.cutoffs.at	Вектор значень відсічок, що потрібно надрукувати уздовж кривої у відповідних точках.

cutoff.label.function	За замовчуванням значення відсічок, що виводяться на кривій та кольоровому ключі, округлюються до двох знаків після коми. Використовуючи цей параметр, можна задати деяке перетворення відсічок перед виведенням (наприклад, округлення або взяття логарифма).
downsampling	За дуже великих розмірів вибірки побудова графіків може бути повільною, а їх розміри надто великими. У цьому разі можна будувати графіки тільки за частиною вибірки. Цей параметр задає константу від 0 до 1, що показує, з якої частини об'єктів потрібно будувати графіки. Якщо значення більше за 1, то графіки будуться по всій вибірці.
add	Якщо TRUE, то криві додаються до вже існуючого графіку, інакше створюється новий графік.

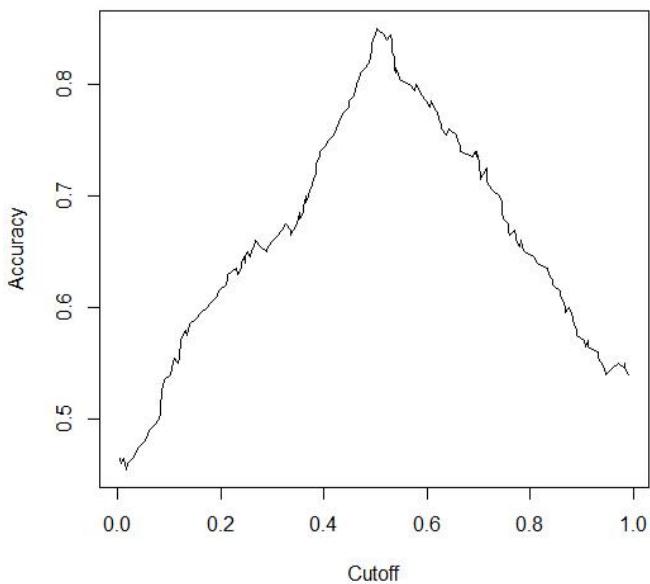
Приклади.

### З одною вибіркою

```
data(ROCR.simple)
pred <- prediction(ROCR.simple$predictions, ROCR.simple$labels)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
```

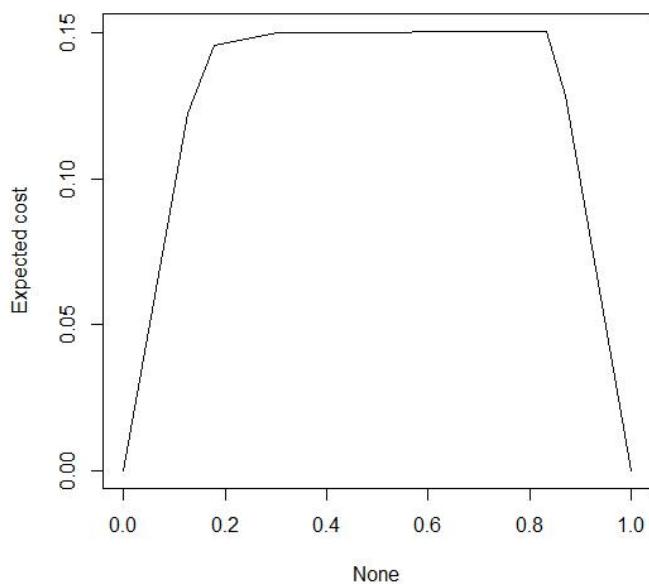


```
perf <- performance(pred, "acc")
plot(perf)
```



```
perf <- performance(pred, "ecost")
```

```
plot(perf)
```



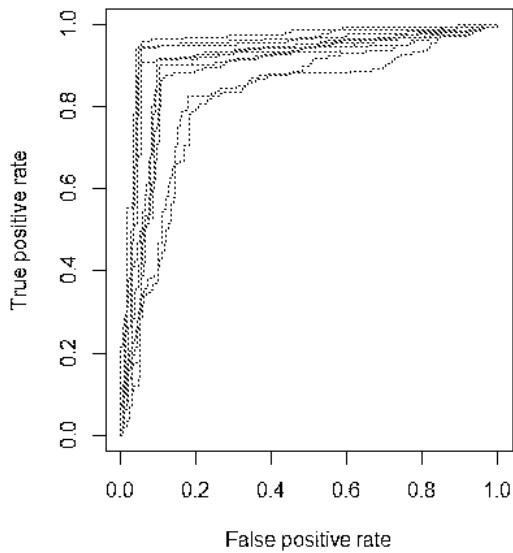
З кількома вибірками

```
data(ROCR.xval)
```

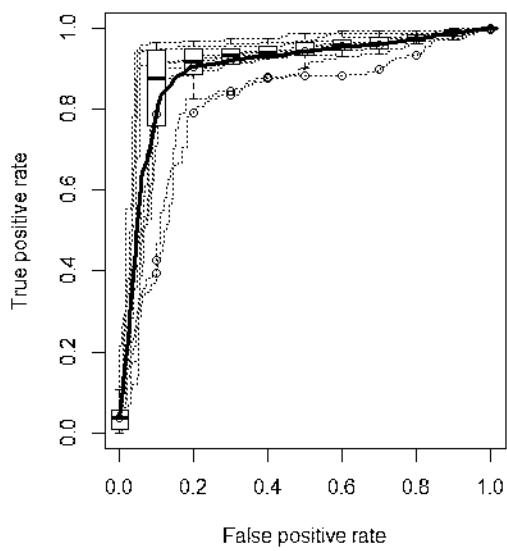
```
pred <- prediction(ROCR.xval$predictions, ROCR.xval$labels)
```

```
perf <- performance(pred, "tpr", "fpr")
```

```
plot(perf,col="black",lty=3)
```



```
> plot(perf,lwd=3,avg="vertical",spread.estimate="boxplot",add=TRUE)
```



## Додаток 6. Зменшення розмірності даних

### Feature Extraction - PCA

Мéтод головнýх компонéнт (МГК, англ. principal component analysis, PCA) – метод факторного аналíзу в статистицí, який використовує ортогональне перетворення множини спостережень з можливо пов'язаними змінними (сущностями, кожна з яких набуває різних числових значень) у множину змінних без лінійної кореляції, які називаються головними компонентами.

МГК – один з основних способів зменшити розмірність даних, втративши найменшу кількість інформації. Винайдений Карлом Пірсоном у 1901 році та доповнений і розширений Гарольдом Готелінгом в 1933. МГК застосовується для наочного подання даних, забезпечення лаконізму моделей, спрощення підрахунку та інтерпретації, стиснення обсягів збереженої інформації. Метод забезпечує максимальну інформативність та мінімальне спотворення геометричної структури початкових даних.

### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('bank.csv', header = TRUE, encoding = 'UNICOD')
f <- f[-1]
```

### Features Encoding & Scaling

```
f$region = as.numeric(factor(f$region, levels = c('TOWN', 'INNER_CITY',
    'SUBURBAN', 'RURAL'), labels = c(1, 2, 3, 4)))
f$sex = as.numeric(factor(f$sex, levels = c('FEMALE', 'MALE'),
    labels = c(1, 2)))
f$married = as.numeric(f$married)
f$car = as.numeric(f$car)
f$mortgage = as.numeric(f$mortgage)
f$delays = as.numeric(f$delays)-1
f[-9] <- as.data.frame(scale(f[-9]))
```

### Delete N/A

```
f <- tidyr::drop_na(f)
cat('there are', nrow(f), 'rows in the f')
```

```
## there are 538 rows in the f
```

## Splitting the dataset into the TRAIN set and TEST set

```
set.seed(123)
library(caTools)
split = sample.split(f$delays, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

## Applying PCA

```
# install.packages('caret')
library(caret)

# install.packages('e1071')
library(e1071)
pca = preProcess(x = f_train[-9], method = 'pca', pcaComp = 2)
f_train_pca = predict(pca, f_train)
f_train_pca = f_train_pca[, c(2, 3, 1)]
f_test_pca = predict(pca, f_test)
f_test_pca = f_test_pca[, c(2, 3, 1)]
```

## Fitting SVM to the PCA-Training set

```
# install.packages('e1071')
library(e1071)
classifier = svm(formula = delays ~ .,
                 data = f_train_pca,
                 type = 'C-classification',
                 kernel = 'linear')
```

## Predicting the Test set results

```
y_pred = predict(classifier, newdata = f_test_pca[-3])
```

## Making the Confusion Matrix

```
cm = table(f_test_pca[, 3], y_pred)
print(cm)

##      y_pred
##      0   1
##  0 78 10
##  1 13 78
```

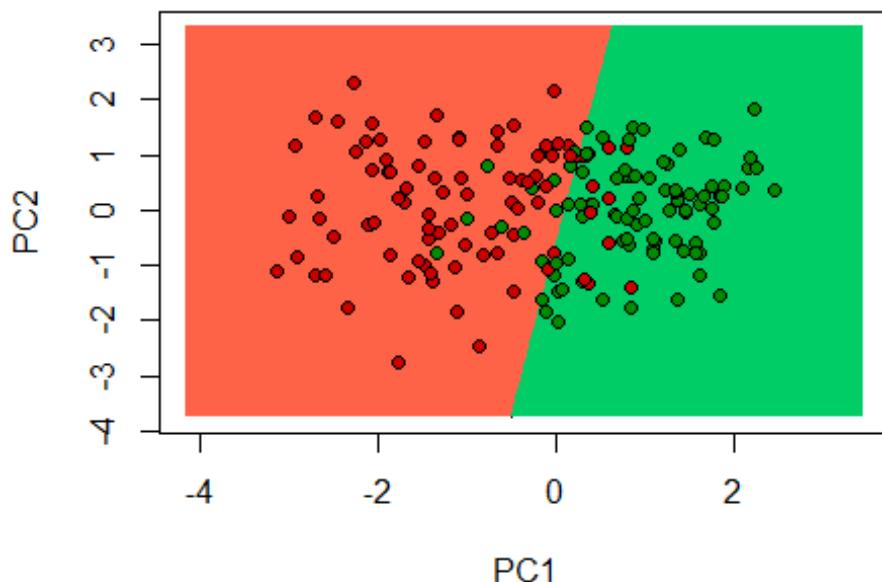
## Visualising the Test set results

```
library(ElemStatLearn)
set = f_test_pca
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('PC1', 'PC2')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3], main = 'SVM (Test set)',
```

```

xlab = 'PC1', ylab = 'PC2',
xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 2, 'deepskyblue', ifelse(
y_grid == 1, 'springgreen3', 'tomato')))
points(set, pch = 21, bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3]
== 1, 'green4', 'red3')))
```

### SVM (Test set)



### Applying Kernel PCA

```

# install.packages('kernlab')
library(kernlab)

kpca = kpca(~., data = f_train[-9], kernel = 'rbfdot', features = 2)
f_train_kpca = as.data.frame(predict(kpca, f_train))
f_train_kpca$delays = f_train$delays
f_test_kpca = as.data.frame(predict(kpca, f_test))
f_test_kpca$delays = f_test$delays
```

### Fitting SVM to the KPCA-Training set

```

# install.packages('e1071')
library(e1071)
classifier = svm(formula = delays ~.,
                  data = f_train_kpca,
                  type = 'C-classification',
                  kernel = 'linear')
```

## Predicting the Test set results

```
y_pred = predict(classifier, newdata = f_test_kpca[-3])
```

## Making the Confusion Matrix

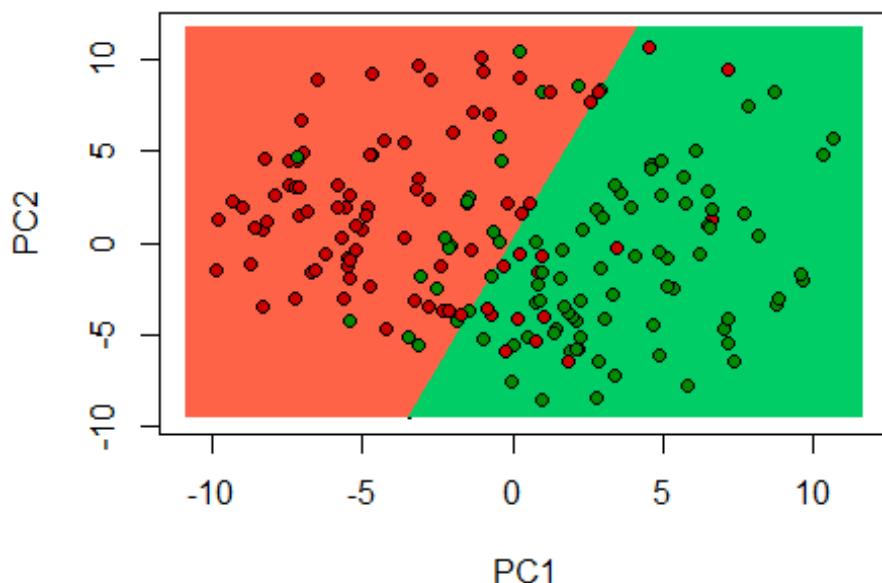
```
cm = table(f_test_kpca[, 3], y_pred)
print(cm)

##      y_pred
##      0  1
##  0 73 15
##  1 21 70
```

## Visualising the Test set results

```
library(ElemStatLearn)
set = f_test_kpca
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('V1', 'V2')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3], main = 'SVM (Test set)',
     xlab = 'PC1', ylab = 'PC2',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 2, 'deepskyblue', ifelse(y_grid == 1, 'springgreen3', 'tomato')))
points(set, pch = 21, bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3')))
```

SVM (Test set)



## Feature Extraction - LDA

Дискримінантний аналіз – різновид багатовимірного аналізу, що використовується для прийняття рішення про те, які змінні розділюють (тобто «дискримінують») певні масиви даних. Основна ідея дискримінантного аналізу полягає в тому, щоб визначити, чи відрізняються сукупності за середнім будь-якої змінної (або їх лінійної комбінації), і потім використовувати цю змінну, щоб передбачити для нових членів їх принадлежність до тієї чи іншої групи.

### Download the data

```
set.seed(123)
#setwd('D:/ML')
f <- read.csv('Wine.csv', header = TRUE, encoding = 'UNICOD')
```

### Features Scaling

```
f[-14] <- as.data.frame(scale(f[-14]))
```

### Splitting the dataset into the TRAIN set and TEST set

```
library(caTools)
split = sample.split(f$Customer_Segment, SplitRatio = 2/3)
f_train = subset(f, split == TRUE)
f_test = subset(f, split == FALSE)
```

### Applying LDA

```
library(MASS)

#It's a supervised algorithm. The number of variables (linear discriminant
s) is equal to the number of classes - 1
lda = lda(formula = Customer_Segment ~ ., data = f_train)
f_train = as.data.frame(predict(lda, f_train))
f_train = f_train[c(5, 6, 1)]
f_test = as.data.frame(predict(lda, f_test))
f_test = f_test[c(5, 6, 1)]
```

### Fitting SVM to the LDA-Training set

```
# install.packages('e1071')
library(e1071)
classifier = svm(formula = class ~.,
                 data = f_train,
                 type = 'C-classification',
                 kernel = 'linear')
```

## Predicting the Test set results

```
y_pred = predict(classifier, newdata = f_test[-3])
```

## Making the Confusion Matrix

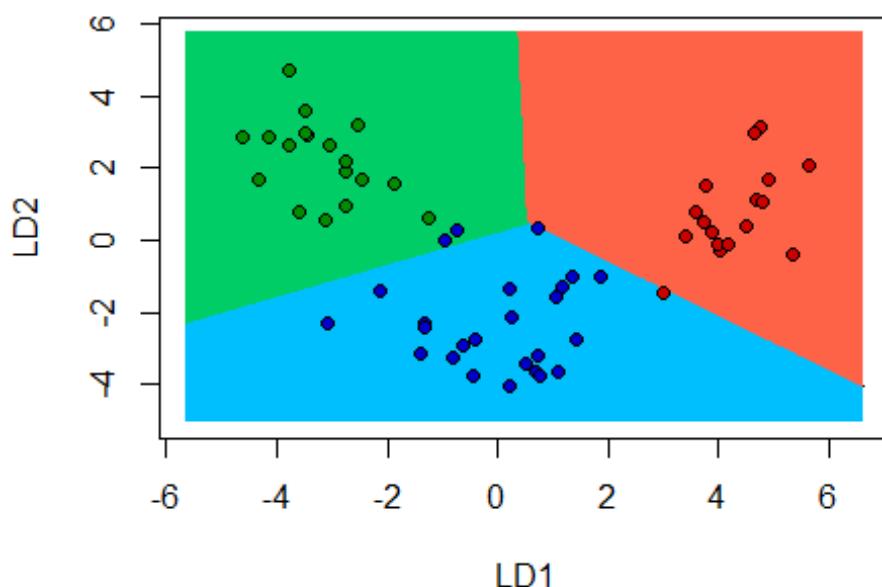
```
cm = table(f_test[, 3], y_pred)
print(cm)
```

```
##      y_pred
##      1 2 3
## 1 18 0 0
## 2 2 23 0
## 3 0 1 16
```

## Visualising the Test set results

```
library(ElemStatLearn)
set = f_test
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('x.LD1', 'x.LD2')
y_grid = predict(classifier, newdata = grid_set)
plot(set[, -3], main = 'SVM (Test set)',
     xlab = 'LD1', ylab = 'LD2',
     xlim = range(X1), ylim = range(X2))
points(grid_set, pch = '.', col = ifelse(y_grid == 2, 'deepskyblue', ifelse(y_grid == 1, 'springgreen3', 'tomato')))
points(set, pch = 21, bg = ifelse(set[, 3] == 2, 'blue3', ifelse(set[, 3] == 1, 'green4', 'red3')))
```

SVM (Test set)



## Додаток 7. PYTHON

### # 01 Data Preprocessing

#### # Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

#### # Importing the dataset

```
df = pd.read_csv('flats.csv', sep=';')
```

```
df
```

	rooms	location	condition	m2	type	price
0	2.0	suburbs	repaired	50	used	35000
1	1.0	center	repaired	37	used	35000
2	3.0	suburbs	repaired	67	used	65000
3	NaN	suburbs	repaired	21	used	15000
4	1.0	suburbs	repaired	82	NaN	60000
...	...	...	...	...	...	...
212	2.0	center	unrepaireded	65	new	46407
213	3.0	center	unrepaireded	84	new	57678
214	3.0	center	unrepaireded	93	new	63513
215	2.0	center	unrepaireded	72	new	51121
216	2.0	center	unrepaireded	74	new	52821

217 rows × 6 columns

```
df[['rooms', 'location', 'condition', 'm2', 'type', 'price']].describe()
```

	rooms	m2	price
count	216.000000	217.000000	217.000000
mean	2.013889	76.331797	82427.451613
std	0.971593	38.019982	82183.660820
min	1.000000	21.000000	1.000000
25%	1.000000	49.000000	39429.000000
50%	2.000000	67.000000	59548.000000
75%	3.000000	91.000000	90000.000000
max	6.000000	280.000000	750000.000000

#### # Cheking NaN

```
df['rooms'].unique(), df['location'].unique(),
df['condition'].unique(), df['m2'].unique(),
df['type'].unique(), df['price'].unique()
(array([ 2.,  1.,  3., nan,  6.,  4.,  5.]),
 array(['suburbs', 'center'], dtype=object),
 array(['repaired', 'unrepaireded'], dtype=object),
 array([ 50,  37,  67,  21,  82,  45,  41,  63,  38,  35, 163,  40,  44,
        53,  65, 215,  78, 143,  68,  57,  54, 100,  60,  70,  84,  83,
        58, 205, 146,  48,  69, 140,  74, 280,  76, 112, 118, 212,  66,
        56,  64,  75,  73,  88,  43, 107,  52,  94,  79,  81,  77,  90,
       117, 126,  97, 105, 137, 114, 123, 128, 189,  86,  49,  55,  46,
       51,  91,  85, 106, 103, 149, 168,  30,  42,  89,  93,  28,  39,
       36, 108, 122, 113,  61,  71,  80,  95, 102, 132, 125, 133, 141,
      127,  33,  72], dtype=int64),
array(['used', nan, 'new'], dtype=object),
array([ 35000,  65000,  15000,  60000,  85000,  48000,  30000,  46000,
       33000,  29000,  25000, 170000,  53000,  58000,  750000,  96000,
```

```
250000, 36000, 52000, 120000, 49000, 40000, 130000, 83000,
63000, 70000, 650000, 90000, 46500, 75000, 280000, 61000,
400000, 1, 200000, 83490, 42000, 51200, 52472, 59548,
57958, 70042, 26801, 27578, 27043, 26422, 27851, 34506,
38018, 38084, 43289, 44196, 62706, 73000, 50000, 39478,
55065, 59677, 56272, 56943, 57850, 58546, 62788, 75078,
28500, 180000, 66301, 62617, 51895, 57851, 37223, 123200,
151815, 160500, 182280, 198650, 176700, 167850, 178350, 185600,
274050, 129300, 27631, 28597, 31883, 33078, 32816, 33272,
43556, 47044, 34400, 43000, 53452, 76342, 64470, 69706,
79226, 76216, 64722, 78386, 70938, 73836, 93800, 127092,
114467, 142695, 139590, 208880, 243310, 42295, 59538, 36060,
28381, 27637, 60730, 89030, 83127, 27471, 30647, 29696,
43182, 62701, 28124, 26832, 34524, 36492, 38907, 46867,
33149, 39429, 33168, 49537, 77003, 66671, 107690, 98670,
116640, 127490, 125840, 110740, 75640, 73983, 32087, 35426,
35530, 36715, 55804, 58280, 74993, 41994, 54732, 63652,
63817, 75776, 160000, 77612, 90112, 106122, 91856, 132920,
132174, 117437, 122867, 113689, 128966, 120056, 134583, 105186,
123250, 154077, 132500, 41000, 58500, 67000, 32500, 43934,
42666, 46407, 57678, 63513, 51121, 52821], dtype=int64))
```

#### # Taking care of missing data

```
from sklearn.impute import SimpleImputer
#numeric
imputer_mean = SimpleImputer(missing_values=np.nan,
strategy='mean').fit(df[['rooms']])
df[['rooms']] = imputer_mean.fit_transform(df[['rooms']])
df[['rooms']] = round(df[['rooms']])
#strings
imputer_freq = SimpleImputer(missing_values=np.nan,
strategy='most_frequent').fit(df[['type']])
df[['type']] = imputer_freq.fit_transform(df[['type']])
df
```

	rooms	location	condition	m2	type	price
0	2.0	suburbs	repaired	50	used	35000
1	1.0	center	repaired	37	used	35000
2	3.0	suburbs	repaired	67	used	65000
3	2.0	suburbs	repaired	21	used	15000
4	1.0	suburbs	repaired	82	new	60000
...	...	...	...	...	...	...
212	2.0	center	unrepaireded	65	new	46407
213	3.0	center	unrepaireded	84	new	57678
214	3.0	center	unrepaireded	93	new	63513
215	2.0	center	unrepaireded	72	new	51121
216	2.0	center	unrepaireded	74	new	52821

217 rows × 6 columns

#### # Encoding categorical data

```
from sklearn.preprocessing import LabelEncoder
df[['location']] =
LabelEncoder().fit_transform(df[['location']])
df[['condition']] =
LabelEncoder().fit_transform(df[['condition']])
df[['type']] = LabelEncoder().fit_transform(df[['type']])
```

df

	rooms	location	condition	m2	type	price
0	2.0	1	0	50	1	35000
1	1.0	0	0	37	1	35000
2	3.0	1	0	67	1	65000
3	2.0	1	0	21	1	15000
4	1.0	1	0	82	0	60000
...	...	...	...	...	...	...
212	2.0	0	1	65	0	46407
213	3.0	0	1	84	0	57678
214	3.0	0	1	93	0	63513
215	2.0	0	1	72	0	51121
216	2.0	0	1	74	0	52821

217 rows × 6 columns

## # Descriptive Statistics

df[['rooms', 'm2', 'price']].describe()

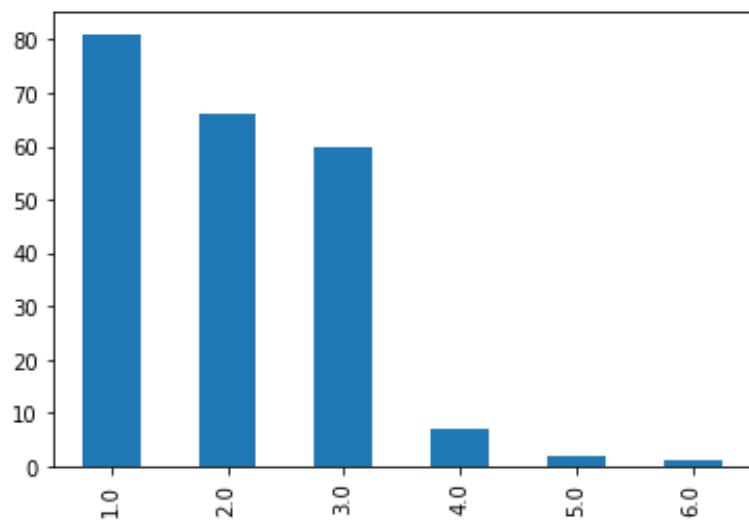
	rooms	m2	price
count	217.000000	217.000000	217.000000
mean	2.013825	76.331797	82427.451613
std	0.969341	38.019982	82183.660820
min	1.000000	21.000000	1.000000
25%	1.000000	49.000000	39429.000000
50%	2.000000	67.000000	59548.000000
75%	3.000000	91.000000	90000.000000
max	6.000000	280.000000	750000.000000

## # Rooms number distribution

distribution = df['rooms'].value\_counts()

distribution.plot(kind='bar')

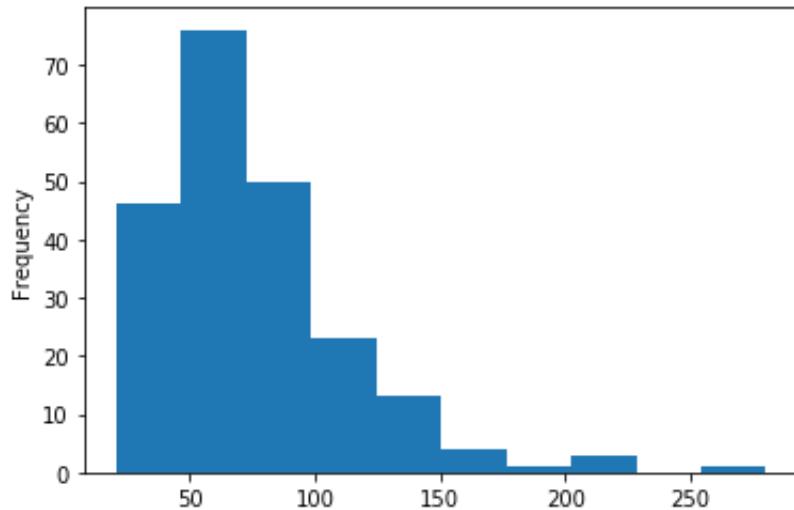
&lt;matplotlib.axes.\_subplots.AxesSubplot at 0x1db3ff9e988&gt;



## # M2 distribution

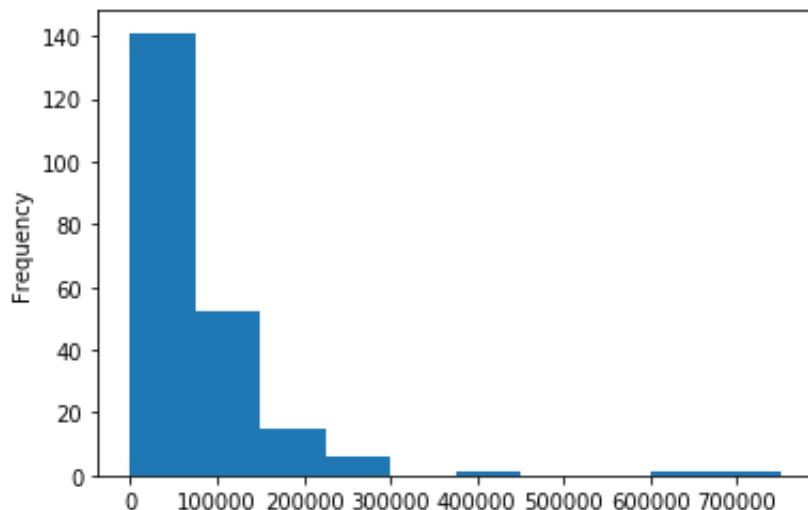
df['m2'].plot(kind = 'hist')

&lt;matplotlib.axes.\_subplots.AxesSubplot at 0x1db4009af48&gt;



#### # Price distribution

```
df['price'].plot(kind = 'hist')
<matplotlib.axes._subplots.AxesSubplot at 0x1db4010f5c8>
```



#### # Outliers

```
rooms = []
for room in df['rooms']:
    if room > round(df['rooms'].mean() + 3 * df['rooms'].std()):
        room = round(df['rooms'].mean() + 3*df['rooms'].std())
    rooms.append(room)
df['rooms'] = rooms

m2 = []
for m in df['m2']:
    if m > round(df['m2'].mean() + 3 * df['m2'].std()):
        m = round(df['m2'].mean() + 3*df['m2'].std())
    m2.append(m)
df['m2'] = m2

prices = []
```

```

for price in df['price']:
    if price > df['price'].mean() + 3 * df['price'].std():
        price = df['price'].mean() + 3*df['price'].std()
    elif price < df['price'].mean() - df['price'].std():
        price = df['price'].mean() - df['price'].std()
    prices.append(price)
df['price'] = prices
df.describe()

```

	rooms	location	condition	m2	type	price
coun	217.00000	217.00000	217.00000	217.00000	217.00000	217.000000
t	0	0	0	0	0	
mean	2.009217	0.267281	0.774194	75.631336	0.202765	78681.728539
std	0.952535	0.443564	0.419079	35.233120	0.402988	59687.878274
min	1.000000	0.000000	0.000000	21.000000	0.000000	243.790793
25%	1.000000	0.000000	1.000000	49.000000	0.000000	39429.000000
50%	2.000000	0.000000	1.000000	67.000000	0.000000	59548.000000
75%	3.000000	1.000000	1.000000	91.000000	0.000000	90000.000000
max	5.000000	1.000000	1.000000	190.000000	1.000000	328978.43407
				0		3

### # Feature Scaling

```
#from sklearn.preprocessing import StandardScaler
```

```
#sc = StandardScaler()
```

```
#dfsc = sc.fit_transform(df)
```

```
#df['rooms'] = dfsc[:,0]
```

```
#df['m2'] = dfsc[:,3]
```

```
#df['price'] = dfsc[:,5]
```

```
df
```

	rooms	location	condition	m2	type	price
0	2.0	1	0	50	1	35000.0
1	1.0	0	0	37	1	35000.0
2	3.0	1	0	67	1	65000.0
3	2.0	1	0	21	1	15000.0
4	1.0	1	0	82	0	60000.0
...	...	...	...	...	...	...
212	2.0	0	1	65	0	46407.0
213	3.0	0	1	84	0	57678.0
214	3.0	0	1	93	0	63513.0
215	2.0	0	1	72	0	51121.0
216	2.0	0	1	74	0	52821.0

217 rows × 6 columns

### # Exporting the dataset

```
df.to_csv('flats_prep.csv', sep=';', index=False)
```

## # 02 Linear Regression

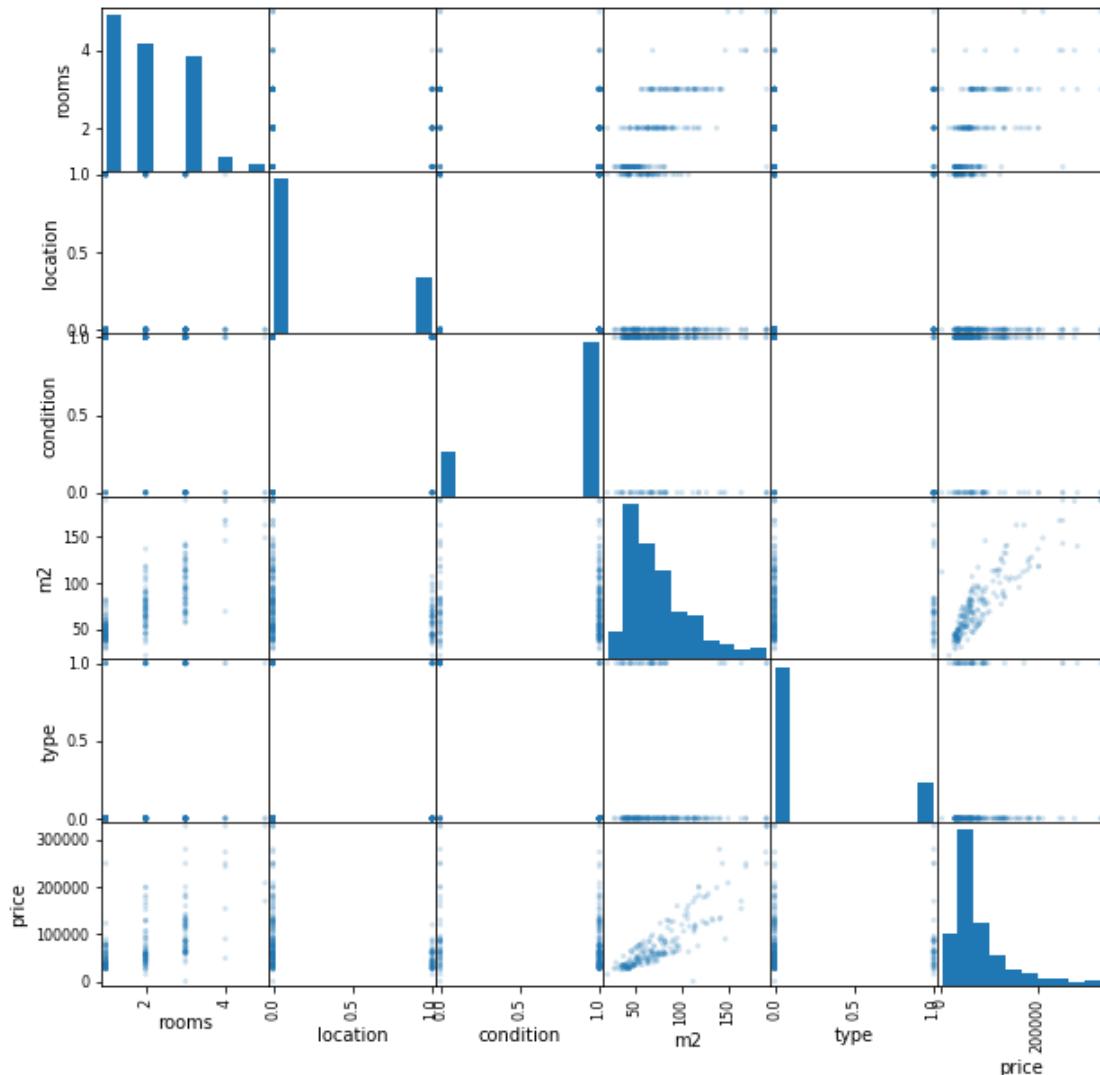
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
df = pd.read_csv('flats_prep.csv', sep=';')
```

### # Cheking correlations

```
df.corr()
   rooms      location     condition       m2      type      price
rooms  1.000000 -0.060645 -0.342692  0.738398  0.272505  0.594447
location -0.060645  1.000000 -0.122117 -0.288718  0.109806 -0.316291
condition -0.342692 -0.122117  1.000000 -0.082482 -0.769335 -0.163575
m2      0.738398 -0.288718 -0.082482  1.000000 -0.043620  0.885215
type    0.272505  0.109806 -0.769335 -0.043620  1.000000  0.020591
price   0.594447 -0.316291 -0.163575  0.885215  0.020591  1.000000
```

```
from pandas.plotting import scatter_matrix
scatter_matrix(df, alpha=0.2, figsize=(10, 10))
plt.show()
```



```

# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 5].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting Simple Linear Regression to the Training set (M2)
from sklearn.linear_model import LinearRegression
sr = LinearRegression().fit(X_train[:, 3:4], y_train)

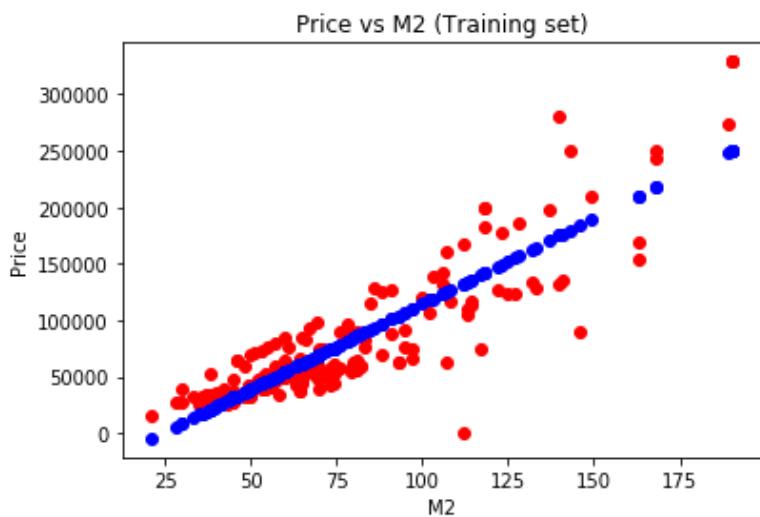
# Getting parameters
sr.coef_, sr.intercept_
(array([1508.56421886]), -36106.702307992935)

# Predicting the Test set results
y_pred = sr.predict(X_test[:, 3:4])

# Coefficient of determination R^2
sr.score(X_train[:, 3:4], y_train), sr.score(X_test[:, 3:4],
y_test)
(0.7872201542559246, 0.7311991996529543)

# Visualising the Training set results
plt.scatter(X_train[:,3], y_train, color = 'red')
plt.plot(X_train[:,3], sr.predict(X_train[:, 3:4]), 'bo')
plt.title('Price vs M2 (Training set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()

```

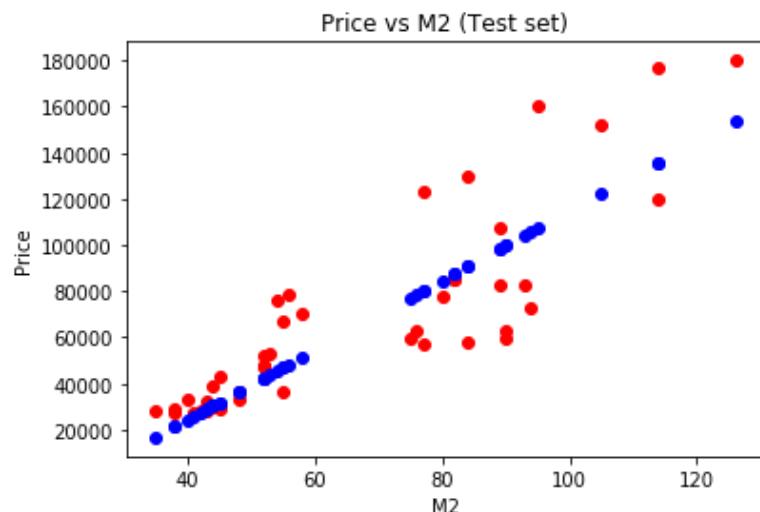


```

# Visualising the Test set results
plt.scatter(X_test[:,3], y_test, color = 'red')
plt.plot(X_test[:,3], sr.predict(X_test[:, 3:4]), 'bo')
plt.title('Price vs M2 (Test set)')
plt.xlabel('M2')
plt.ylabel('Price')

```

```
plt.show()
```



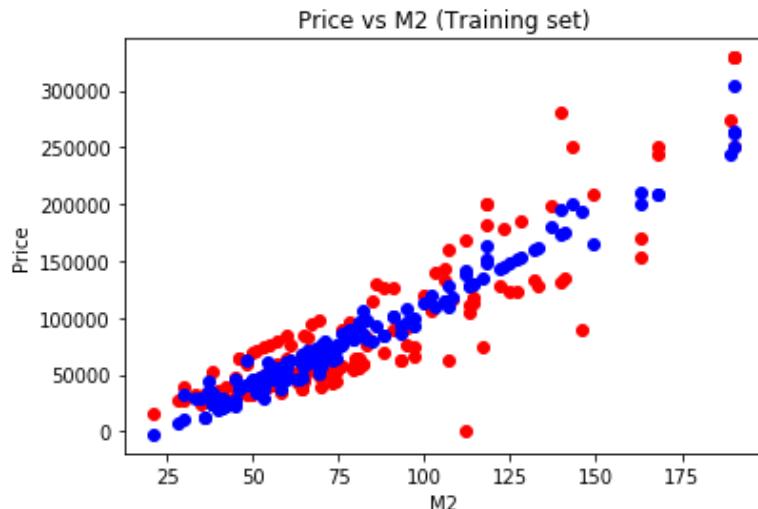
```
# Fitting Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
mr = LinearRegression().fit(X_train, y_train)

# Getting parameters
mr.coef_, mr.intercept_
(array([-11699.52495304, -7845.14536414, -16008.41803329,
       1695.68033309, 6254.7393622 ]),
 -13108.21121704494)

# Predicting the Test set results
y_pred = mr.predict(X_test)

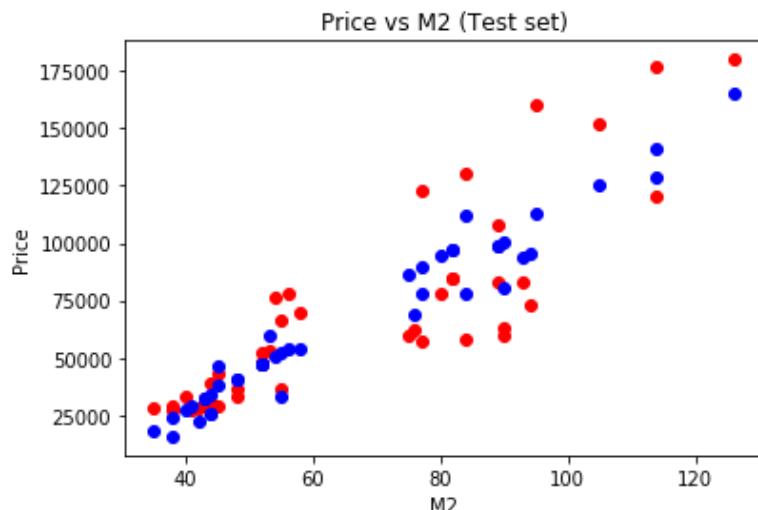
# Coefficient of determination R^2
mr.score(X_train, y_train), mr.score(X_test, y_test)
(0.8106927187926689, 0.8208439234294531)

# Visualising the Training set results
plt.scatter(X_train[:,3], y_train, color = 'red')
plt.plot(X_train[:,3], mr.predict(X_train), 'bo')
plt.title('Price vs M2 (Training set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



#### # Visualising the Test set results

```
plt.scatter(X_test[:,3], y_test, color = 'red')
plt.plot(X_test[:,3], mr.predict(X_test), 'bo')
plt.title('Price vs M2 (Test set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



#### # Fitting Polynomial Regression to the dataset

```
from sklearn.preprocessing import PolynomialFeatures
X_train_p = PolynomialFeatures().fit_transform(X_train[:, 3:4])
X_test_p = PolynomialFeatures().fit_transform(X_test[:, 3:4])
pr = LinearRegression().fit(X_train_p[:,1:], y_train)
```

#### # Getting parameters

```
pr.coef_, pr.intercept_
(array([158.05071132, -6.75702175]), 18785.27193613172)
```

#### # Predicting the Test set results

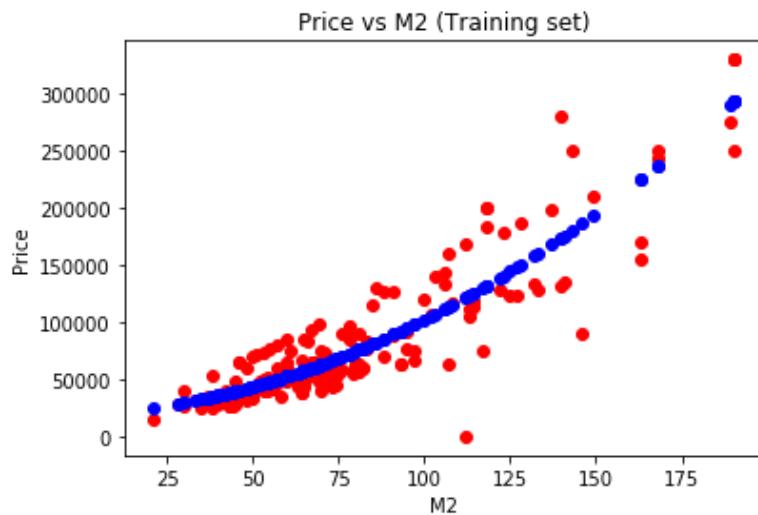
```
y_pred = pr.predict(X_test_p[:,1:])
```

#### # Coefficient of determination R^2

```
pr.score(X_train_p[:,1:], y_train), pr.score(X_test_p[:,1:],  
y_test)  
(0.8218549974270769, 0.7080453786249812)
```

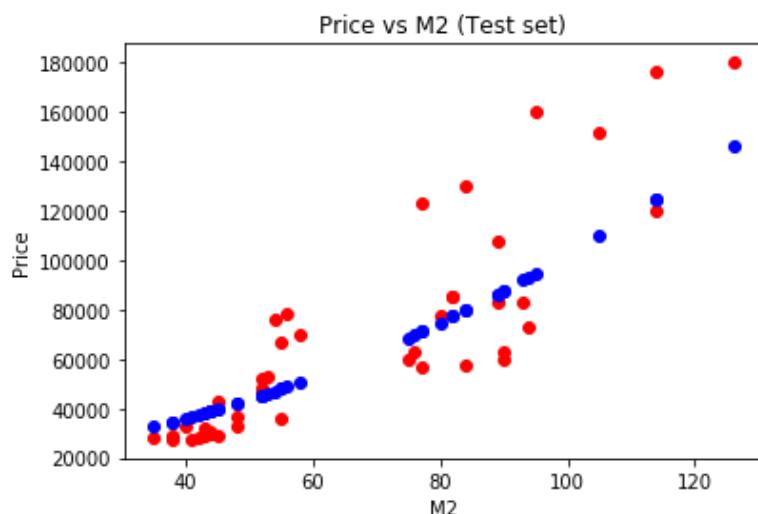
**# Visualising the Training set results**

```
plt.scatter(X_train[:,3], y_train, color = 'red')  
plt.plot(X_train[:,3], pr.predict(X_train_p[:,1:]), 'bo')  
plt.title('Price vs M2 (Training set)')  
plt.xlabel('M2')  
plt.ylabel('Price')  
plt.show()
```



**# Visualising the Test set results**

```
plt.scatter(X_test[:,3], y_test, color = 'red')  
plt.plot(X_test[:,3], pr.predict(X_test_p[:,1:]), 'bo')  
plt.title('Price vs M2 (Test set)')  
plt.xlabel('M2')  
plt.ylabel('Price')  
plt.show()
```



**# Backward Elimination with p-values**

```
import statsmodels.api as sm  
def backwardElimination(x, sl):  
    numVars = len(x[0])
```

```

for i in range(0, numVars):
    regressor_OLS = sm.OLS(y, x).fit()
    maxVar = max(regressor_OLS.pvalues).astype(float)
    if maxVar > sl:
        for j in range(0, numVars - i):
            if (regressor_OLS.pvalues[j].astype(float) ==
maxVar):
                x = np.delete(x, j, 1)
    regressor_OLS.summary()
return x

SL = 0.05
X_opt = X_train[:, [0, 1, 2, 3, 4]]
y = y_train
X_Modeled = backwardElimination(X_opt, SL)

# Fitting Optimized Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
omr = LinearRegression().fit(X_train[:, 0:4], y_train)

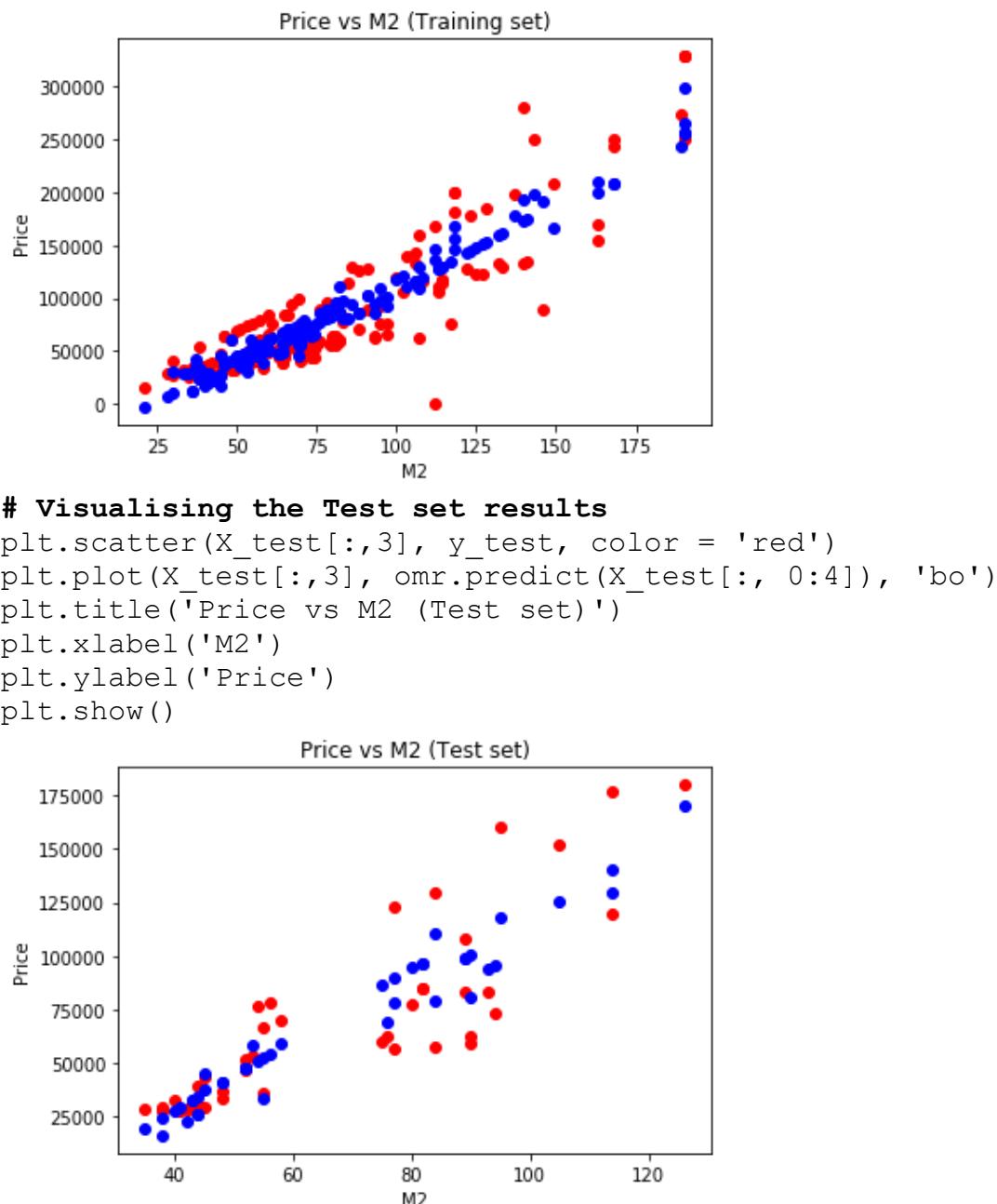
# Getting parameters
omr.coef_, omr.intercept_
(array([-11060.65680349, -8092.5842503, -20423.7590526,
1674.95218774]),
 -7988.89098065703)

# Predicting the Test set results
y_pred = omr.predict(X_test[:, 0:4])

# Coefficient of determination R^2
omr.score(X_train[:, 0:4], y_train), omr.score(X_test[:, 0:4],
y_test)
(0.8100612955370429, 0.8287517619676935)

# Visualising the Training set results
plt.scatter(X_train[:,3], y_train, color = 'red')
plt.plot(X_train[:,3], omr.predict(X_train[:, 0:4]), 'bo')
plt.title('Price vs M2 (Training set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()

```



## # 03 Regression Tree &amp; Random Forest

```

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
df = pd.read_csv('flats_prep.csv', sep=';')

# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 5].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

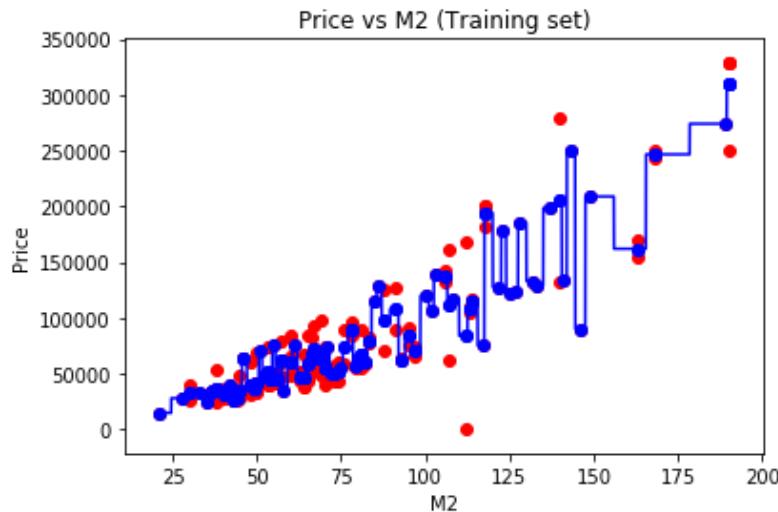
# Fitting Tree to the Training set (M2)
from sklearn.tree import DecisionTreeRegressor
sdt = DecisionTreeRegressor().fit(X_train[:, 3:4], y_train)

# Predicting the Test set results
y_pred = sdt.predict(X_test[:, 3:4])

# Coefficient of determination R^2
sdt.score(X_train[:, 3:4], y_train), sdt.score(X_test[:, 3:4],
y_test)
(0.929393214897912, 0.5834460431627706)

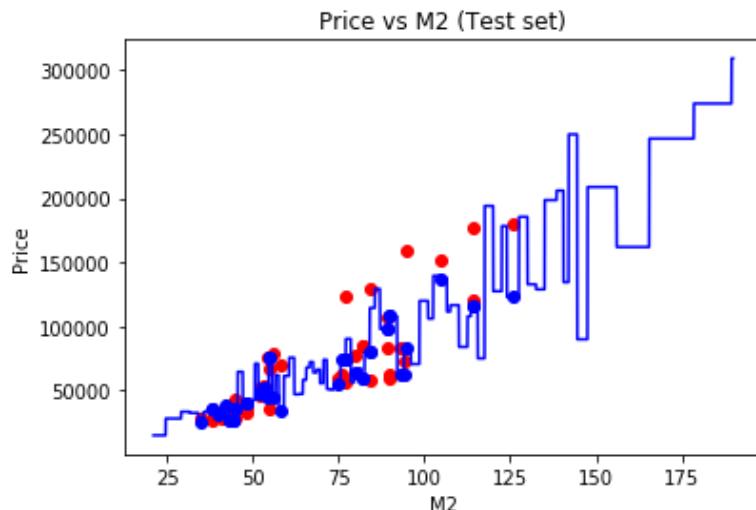
# Visualising the Training set results
X_grid = np.arange(min(X[:, 3:4]), max(X[:, 3:4]), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.plot(X_grid, sdt.predict(X_grid), color = 'blue')
plt.scatter(X_train[:,3], y_train, color = 'red')
plt.plot(X_train[:,3], sdt.predict(X_train[:, 3:4]), 'bo')
plt.title('Price vs M2 (Training set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()

```



**# Visualising the Test set results**

```
X_grid = np.arange(min(X[:, 3:4]), max(X[:, 3:4]), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.plot(X_grid, sdt.predict(X_grid), color = 'blue')
plt.scatter(X_test[:,3], y_test, color = 'red')
plt.plot(X_test[:,3], sdt.predict(X_test[:, 3:4]), 'bo')
plt.title('Price vs M2 (Test set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



**# Fitting Tree to the Training set**

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor().fit(X_train, y_train)
```

**# Predicting the Test set results**

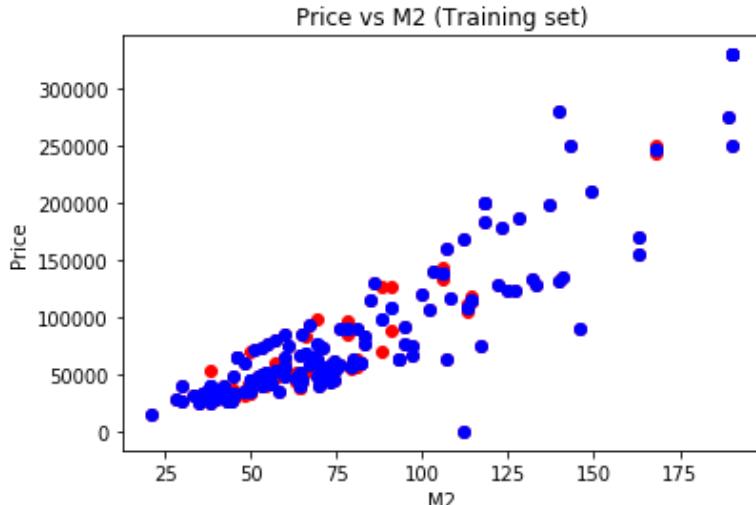
```
y_pred = dt.predict(X_test)
```

**# Coefficient of determination R^2**

```
dt.score(X_train, y_train), dt.score(X_test, y_test)
(0.9921189919121658, 0.7390017339788906)
```

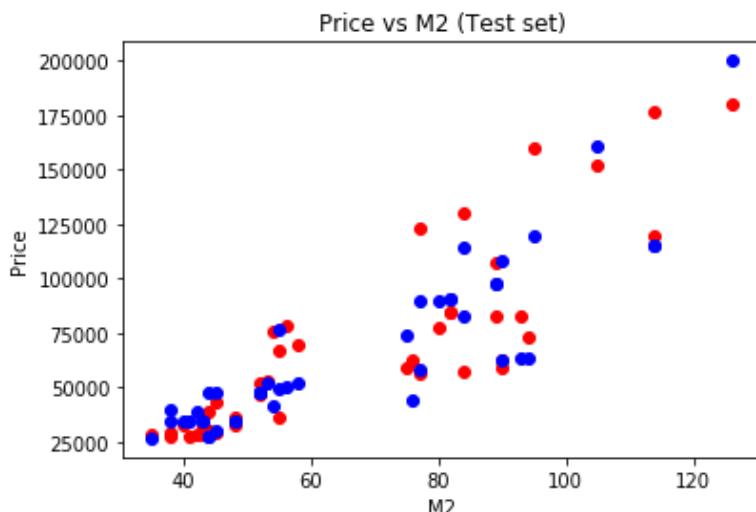
**# Visualising the Training set results**

```
plt.scatter(X_train[:,3], y_train, color = 'red')
plt.plot(X_train[:,3], dt.predict(X_train), 'bo')
plt.title('Price vs M2 (Training set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



#### # Visualising the Test set results

```
plt.scatter(X_test[:,3], y_test, color = 'red')
plt.plot(X_test[:,3], dt.predict(X_test), 'bo')
plt.title('Price vs M2 (Test set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



#### # Fitting Random Forest to the Training set

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 10, random_state = 0).fit(X_train, y_train)
```

#### # Predicting the Test set results

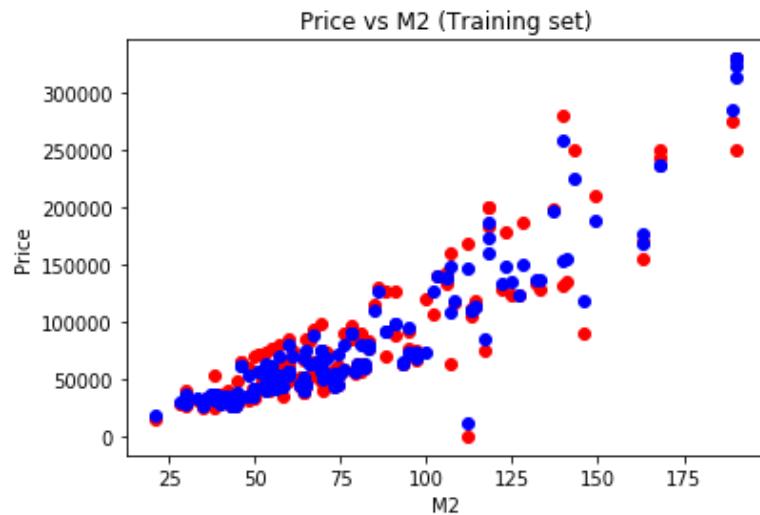
```
y_pred = rf.predict(X_test)
```

#### # Coefficient of determination R^2

```
rf.score(X_train, y_train), rf.score(X_test, y_test)
(0.9629307850743382, 0.7069710407747708)
```

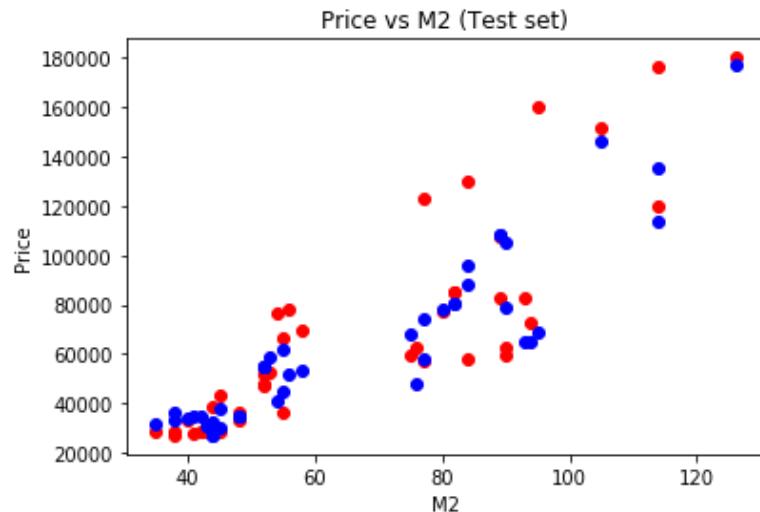
**# Visualising the Training set results**

```
plt.scatter(X_train[:,3], y_train, color = 'red')
plt.plot(X_train[:,3], rf.predict(X_train), 'bo')
plt.title('Price vs M2 (Training set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



**# Visualising the Test set results**

```
plt.scatter(X_test[:,3], y_test, color = 'red')
plt.plot(X_test[:,3], rf.predict(X_test), 'bo')
plt.title('Price vs M2 (Test set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



## # 04 Logistic Regression

## # Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

## # Importing the dataset

```
df = pd.read_csv('bank.csv')
```

```
df
```

	<b>id</b>	<b>age</b>	<b>sex</b>	<b>region</b>	<b>income</b>	<b>married</b>	<b>children</b>	<b>car</b>	<b>mortgage</b>	<b>delays</b>
0	ID12101	48	FEMALE	INNER_CITY	17546.00	NO	1	NO	NO	YES
1	ID12102	40	MALE	TOWN	30085.10	YES	3	YES	YES	NO
2	ID12103	51	FEMALE	INNER_CITY	16575.40	YES	0	YES	NO	YES
3	ID12104	23	FEMALE	TOWN	20375.40	YES	3	NO	NO	YES
4	ID12105	57	FEMALE	RURAL	50576.30	YES	0	NO	NO	NO
...	...	...	...	...	...	...	...	...	...	...
595	ID12696	61	FEMALE	INNER_CITY	47025.00	NO	2	YES	YES	NO
596	ID12697	30	FEMALE	INNER_CITY	9672.25	YES	0	YES	NO	YES
597	ID12698	31	FEMALE	TOWN	15976.30	YES	0	YES	NO	YES
598	ID12699	29	MALE	INNER_CITY	14711.80	YES	0	NO	YES	YES
599	ID12700	38	MALE	TOWN	26671.60	NO	0	YES	YES	NO

600 rows × 10 columns

## # Descriptive Statistics

```
df.describe()
```

	<b>age</b>	<b>income</b>	<b>children</b>
<b>count</b>	600.000000	600.000000	600.000000
<b>mean</b>	42.395000	27524.031217	1.011667
<b>std</b>	14.424947	12899.468246	1.056752
<b>min</b>	18.000000	5014.210000	0.000000
<b>25%</b>	30.000000	17264.500000	0.000000
<b>50%</b>	42.000000	24925.300000	1.000000
<b>75%</b>	55.250000	36172.675000	2.000000
<b>max</b>	67.000000	63130.100000	3.000000

## # Encoding categorical data

```
df =
df[['age','sex','income','married','children','car','mortgage',
,'delays']]
from sklearn.preprocessing import LabelEncoder
df.loc[:, 'sex'] = LabelEncoder().fit_transform(df[['sex']])
df.loc[:, 'married'] =
LabelEncoder().fit_transform(df[['married']])
df.loc[:, 'car'] = LabelEncoder().fit_transform(df[['car']])
df.loc[:, 'mortgage'] =
LabelEncoder().fit_transform(df[['mortgage']])
df.loc[:, 'delays'] =
LabelEncoder().fit_transform(df[['delays']])
df
```

	<b>age</b>	<b>sex</b>	<b>income</b>	<b>married</b>	<b>children</b>	<b>car</b>	<b>mortgage</b>	<b>delays</b>
0	48	0	17546.00	0	1	0	0	1
1	40	1	30085.10	1	3	1	1	0
2	51	0	16575.40	1	0	1	0	1
3	23	0	20375.40	1	3	0	0	1
4	57	0	50576.30	1	0	0	0	0

...	...	...	...	...	...	...	...	...
595	61	0	47025.00	0	2	1	1	0
596	30	0	9672.25	1	0	1	0	1
597	31	0	15976.30	1	0	1	0	1
598	29	1	14711.80	1	0	0	1	1
599	38	1	26671.60	0	0	1	1	0

600 rows × 8 columns

### # Feature Scaling

```
from sklearn.preprocessing import StandardScaler
dfsc = StandardScaler().fit_transform(df)
df.loc[:, 'age'] = dfsc[:, 0]
df.loc[:, 'income'] = dfsc[:, 2]
df.loc[:, 'children'] = dfsc[:, 4]
df
```

	age	sex	income	married	children	car	mortgage	delays
0	0.388887	0	-0.774168	0	-0.011049	0	0	1
1	-0.166170	1	0.198706	1	1.883121	1	1	0
2	0.597034	0	-0.849474	1	-0.958135	1	0	1
3	-1.345667	0	-0.554643	1	1.883121	0	0	1
4	1.013327	0	1.788562	1	-0.958135	0	0	0
...	...	...	...	...	...	...	...	...
595	1.290856	0	1.513027	0	0.936036	1	1	0
596	-0.859992	0	-1.385071	1	-0.958135	1	0	1
597	-0.790610	0	-0.895957	1	-0.958135	1	0	1
598	-0.929374	1	-0.994066	1	-0.958135	0	1	1
599	-0.304935	1	-0.066138	0	-0.958135	1	1	0

600 rows × 8 columns

### # Exporting the dataset

```
df.to_csv('bank_prep.csv', index=False)
```

```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

### # Fitting Logistic Regression to the Training set (2 variables)

```
from sklearn.linear_model import LogisticRegression
slr = LogisticRegression(random_state = 0).fit(X_train,
y_train)
```

### # Predicting the Test set results

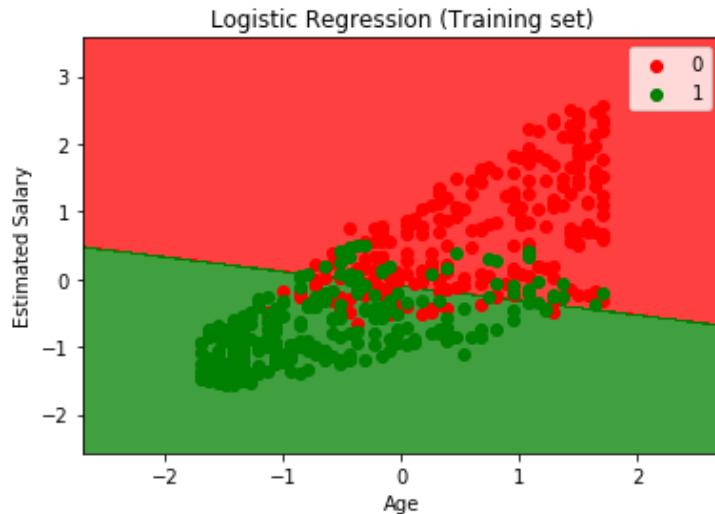
```
y_pred = slr.predict(X_test)
```

### # Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

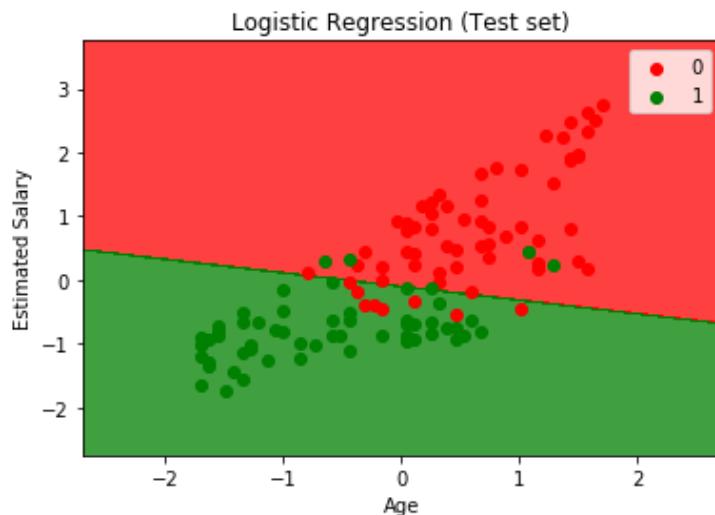
```
[ [55  8]
 [ 5 52] ]
```

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, slr.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, slr.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red',
'green')))
```

```
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()
```



```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting Logistic Regression to the Training set (all variables)
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state = 0).fit(X_train,
y_train)

# Predicting the Test set results
y_pred = lr.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[54  9]
 [ 6 51]]
```

## # 05 K-Nearest Neighbors (K-NN)

```
# Importing the libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Importing the dataset
```

```
df = pd.read_csv('bank_prep.csv')
df
```

	age	sex	income	married	children	car	mortgage	delays
0	0.388887	0	-0.774168	0	-0.011049	0	0	1
1	-0.166170	1	0.198706	1	1.883121	1	1	0
2	0.597034	0	-0.849474	1	-0.958135	1	0	1
3	-1.345667	0	-0.554643	1	1.883121	0	0	1
4	1.013327	0	1.788562	1	-0.958135	0	0	0
...	...	...	...	...	...	...	...	...
595	1.290856	0	1.513027	0	0.936036	1	1	0
596	-0.859992	0	-1.385071	1	-0.958135	1	0	1
597	-0.790610	0	-0.895957	1	-0.958135	1	0	1
598	-0.929374	1	-0.994066	1	-0.958135	0	1	1
599	-0.304935	1	-0.066138	0	-0.958135	1	1	0

600 rows × 8 columns

```
# Splitting the dataset into the Training set and Test set
```

```
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

```
# Fitting K-NN to the Training set (2 variables)
```

```
from sklearn.neighbors import KNeighborsClassifier
sknn = KNeighborsClassifier(n_neighbors = 5, metric =
'minkowski', p = 2).fit(X_train, y_train)
```

```
# Predicting the Test set results
```

```
y_pred = sknn.predict(X_test)
```

```
# Making the Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[52 11]
 [ 7 50]]
```

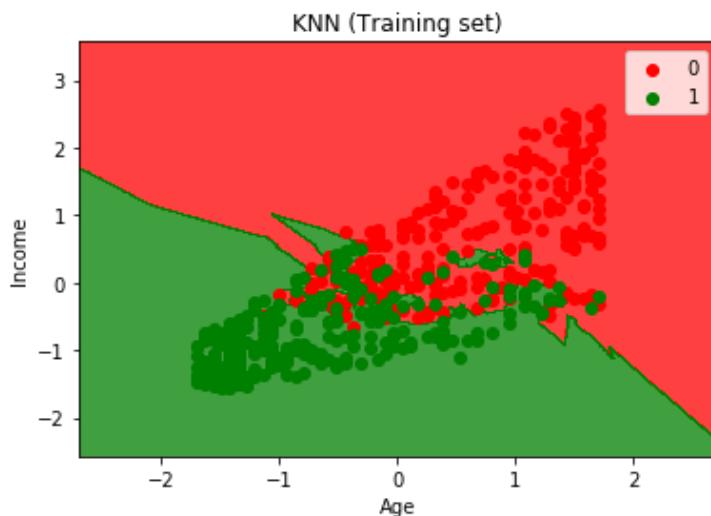
```
# Visualising the Training set results
```

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
```

```

        np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, sknn.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('KNN (Training set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()

```



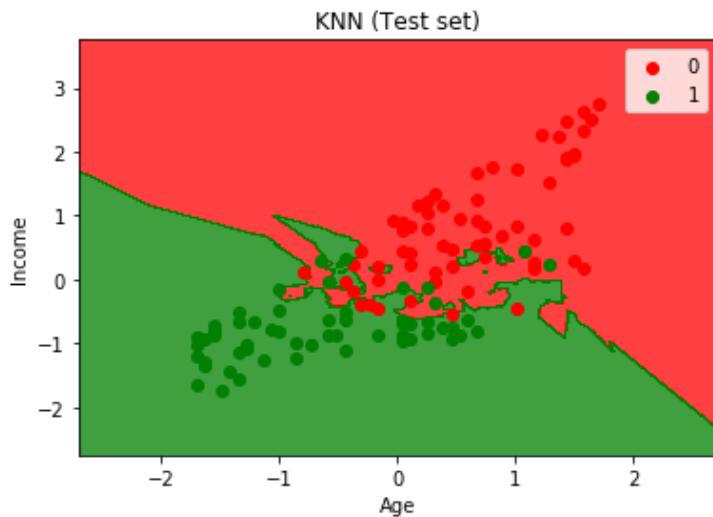
#### # Visualising the Test set results

```

from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, sknn.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('KNN (Test set)')
plt.xlabel('Age')

```

```
plt.ylabel('Income')
plt.legend()
plt.show()
```



```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5, metric =
'minkowski', p = 2).fit(X_train, y_train)

# Predicting the Test set results
y_pred = knn.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[54  9]
 [ 5 52]]
```

## # 06 Support Vector Machine (SVM), Linear Kernel

### # Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

### # Importing the dataset

```
df = pd.read_csv('bank_prep.csv')
df
```

	age	sex	income	married	children	car	mortgage	delays
0	0.388887	0	-0.774168	0	-0.011049	0	0	1
1	-0.166170	1	0.198706	1	1.883121	1	1	0
2	0.597034	0	-0.849474	1	-0.958135	1	0	1
3	-1.345667	0	-0.554643	1	1.883121	0	0	1
4	1.013327	0	1.788562	1	-0.958135	0	0	0
...	...	...	...	...	...	...	...	...
595	1.290856	0	1.513027	0	0.936036	1	1	0
596	-0.859992	0	-1.385071	1	-0.958135	1	0	1
597	-0.790610	0	-0.895957	1	-0.958135	1	0	1
598	-0.929374	1	-0.994066	1	-0.958135	0	1	1
599	-0.304935	1	-0.066138	0	-0.958135	1	1	0

600 rows × 8 columns

### # Splitting the dataset into the Training set and Test set

```
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

### # Fitting SVM to the Training set (2 variables)

```
from sklearn.svm import SVC
ssvm = SVC(kernel = 'linear', random_state = 0).fit(X_train,
y_train)
```

### # Predicting the Test set results

```
y_pred = ssvm.predict(X_test)
```

### # Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[55  8]
 [ 6 51]]
```

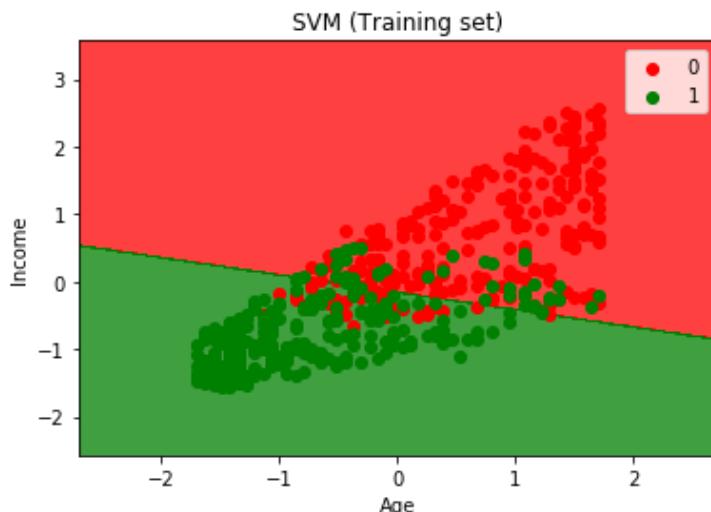
### # Visualising the Training set results

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
```

```

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, ssvm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
              alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()

```

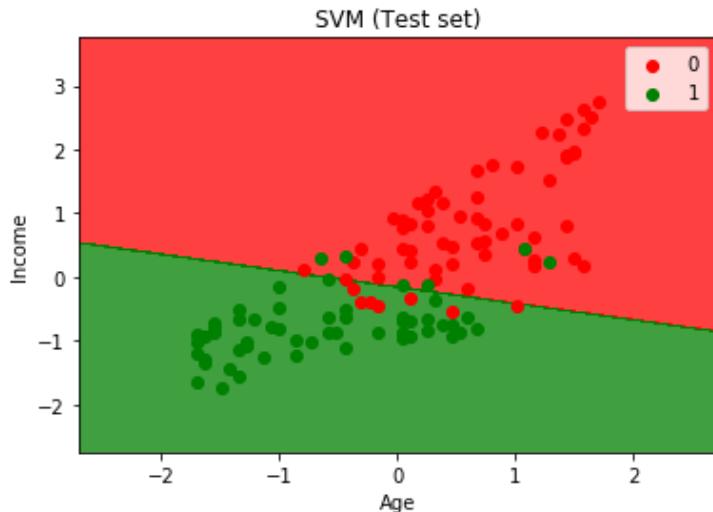


```

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, ssvm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
              alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)

```

```
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()
```



```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting SVM to the Training set
from sklearn.svm import SVC
svm = SVC(kernel = 'linear', random_state = 0).fit(X_train,
y_train)

# Predicting the Test set results
y_pred = svm.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[56  7]
 [ 6 51]]
```

## # 07 Kernel SVM

```
# Importing the libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Importing the dataset
```

```
df = pd.read_csv('bank_prep.csv')
df
```

	age	sex	income	married	children	car	mortgage	delays
0	0.388887	0	-0.774168	0	-0.011049	0	0	1
1	-0.166170	1	0.198706	1	1.883121	1	1	0
2	0.597034	0	-0.849474	1	-0.958135	1	0	1
3	-1.345667	0	-0.554643	1	1.883121	0	0	1
4	1.013327	0	1.788562	1	-0.958135	0	0	0
...	...	...	...	...	...	...	...	...
595	1.290856	0	1.513027	0	0.936036	1	1	0
596	-0.859992	0	-1.385071	1	-0.958135	1	0	1
597	-0.790610	0	-0.895957	1	-0.958135	1	0	1
598	-0.929374	1	-0.994066	1	-0.958135	0	1	1
599	-0.304935	1	-0.066138	0	-0.958135	1	1	0

600 rows × 8 columns

```
# Splitting the dataset into the Training set and Test set
```

```
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

```
# Fitting Kernel SVM to the Training set (2 variables)
```

```
from sklearn.svm import SVC
ssvm = SVC(kernel = 'rbf', random_state = 0).fit(X_train,
y_train)
```

```
# Predicting the Test set results
```

```
y_pred = ssvm.predict(X_test)
```

```
# Making the Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[55  8]
 [ 7 50]]
```

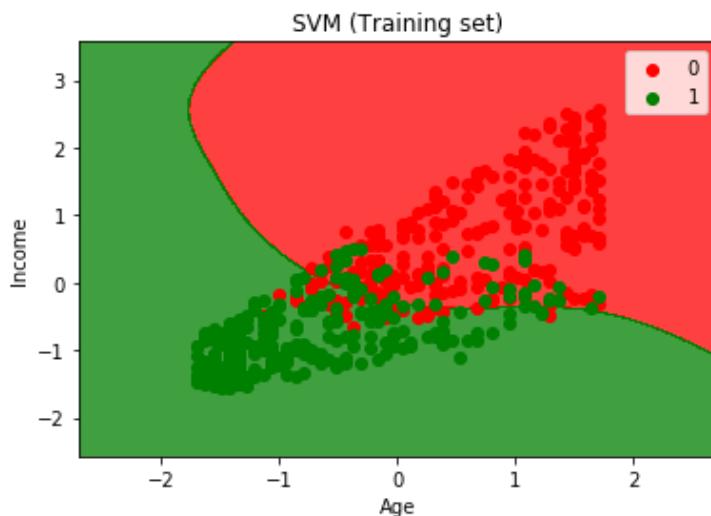
```
# Visualising the Training set results
```

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
```

```

        np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, ssvm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()

```



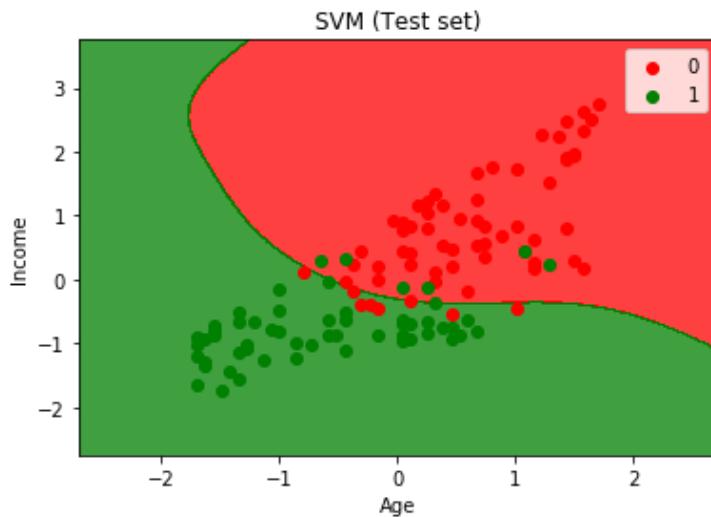
#### # Visualising the Test set results

```

from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, ssvm.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('SVM (Test set)')
plt.xlabel('Age')

```

```
plt.ylabel('Income')
plt.legend()
plt.show()
```



```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting Kernel SVM to the Training set
from sklearn.svm import SVC
svm = SVC(kernel = 'rbf', random_state = 0).fit(X_train,
y_train)

# Predicting the Test set results
y_pred = svm.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[59  4]
 [ 6 51]]
```

## # 08 Naive Bayes

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Importing the dataset
df = pd.read_csv('bank_prep.csv')
df
```

	age	sex	income	married	children	car	mortgage	delays
0	0.388887	0	-0.774168	0	-0.011049	0	0	1
1	-0.166170	1	0.198706	1	1.883121	1	1	0
2	0.597034	0	-0.849474	1	-0.958135	1	0	1
3	-1.345667	0	-0.554643	1	1.883121	0	0	1
4	1.013327	0	1.788562	1	-0.958135	0	0	0
...	...	...	...	...	...	...	...	...
595	1.290856	0	1.513027	0	0.936036	1	1	0
596	-0.859992	0	-1.385071	1	-0.958135	1	0	1
597	-0.790610	0	-0.895957	1	-0.958135	1	0	1
598	-0.929374	1	-0.994066	1	-0.958135	0	1	1
599	-0.304935	1	-0.066138	0	-0.958135	1	1	0

600 rows × 8 columns

```
# Splitting the dataset into the Training set and Test set
```

```
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

```
# Fitting Naive Bayes to the Training set (2 variables)
```

```
from sklearn.naive_bayes import GaussianNB
snb = GaussianNB().fit(X_train, y_train)
```

```
# Predicting the Test set results
```

```
y_pred = snb.predict(X_test)
```

```
# Making the Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[54  9]
 [ 4 53]]
```

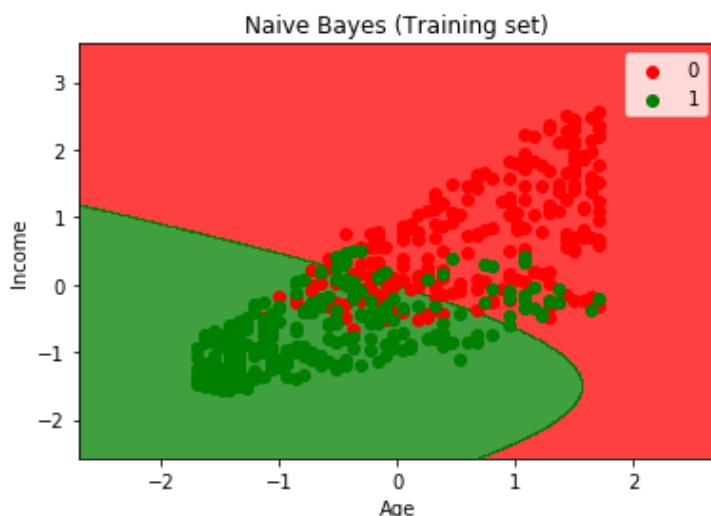
```
# Visualising the Training set results
```

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
```

```

plt.contourf(X1, X2, snb.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()

```



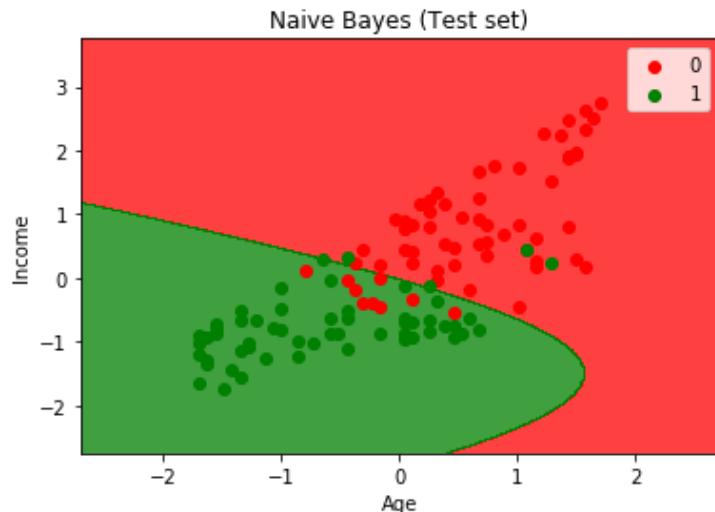
#### # Visualising the Test set results

```

from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, snb.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('Naive Bayes (Test set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()

```

```
plt.show()
```



```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB().fit(X_train, y_train)

# Predicting the Test set results
y_pred = nb.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[54  9]
 [ 4 53]]
```

## # 09 Classification Tree

```
# Importing the libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Importing the dataset
```

```
df = pd.read_csv('bank_prep.csv')
df
```

	age	sex	income	married	children	car	mortgage	delays
0	0.388887	0	-0.774168	0	-0.011049	0	0	1
1	-0.166170	1	0.198706	1	1.883121	1	1	0
2	0.597034	0	-0.849474	1	-0.958135	1	0	1
3	-1.345667	0	-0.554643	1	1.883121	0	0	1
4	1.013327	0	1.788562	1	-0.958135	0	0	0
...	...	...	...	...	...	...	...	...
595	1.290856	0	1.513027	0	0.936036	1	1	0
596	-0.859992	0	-1.385071	1	-0.958135	1	0	1
597	-0.790610	0	-0.895957	1	-0.958135	1	0	1
598	-0.929374	1	-0.994066	1	-0.958135	0	1	1
599	-0.304935	1	-0.066138	0	-0.958135	1	1	0

600 rows × 8 columns

```
# Splitting the dataset into the Training set and Test set
```

```
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

```
# Fitting Classification Tree to the Training set (2
variables)
```

```
from sklearn.tree import DecisionTreeClassifier
sct = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0).fit(X_train, y_train)
```

```
# Predicting the Test set results
```

```
y_pred = sct.predict(X_test)
```

```
# Making the Confusion Matrix
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[49 14]
```

```
[ 7 50]]
```

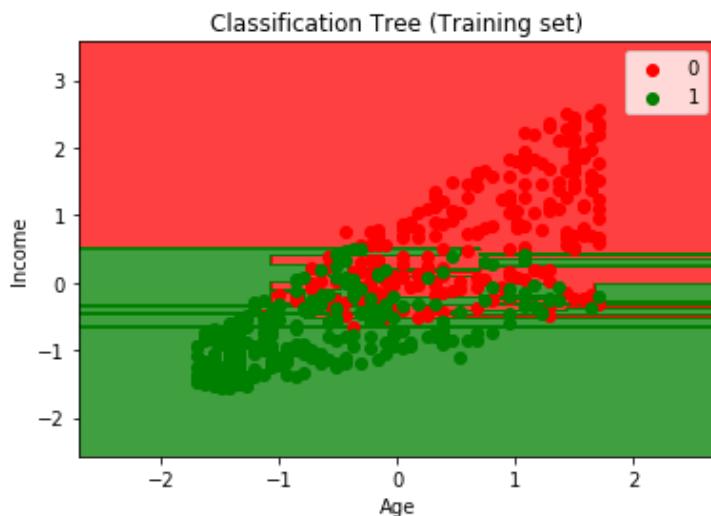
```
# Visualising the Training set results
```

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
```

```

np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, sct.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('Classification Tree (Training set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()

```

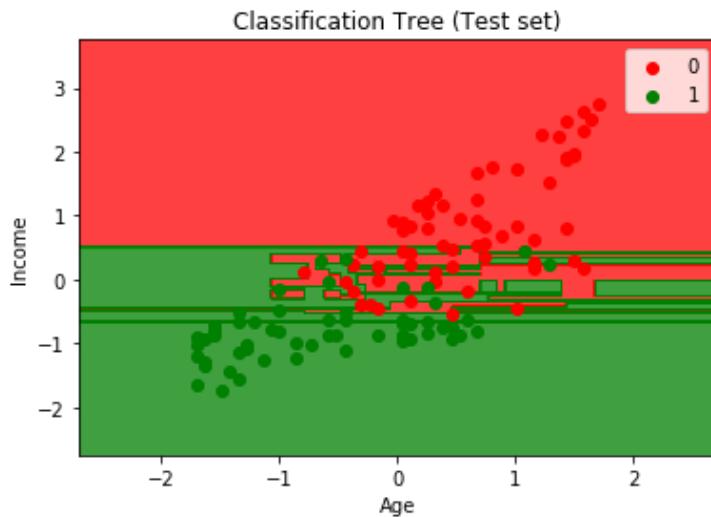


```

# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, sct.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('Classification Tree (Test set)')
plt.xlabel('Age')

```

```
plt.ylabel('Income')
plt.legend()
plt.show()
```



```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting Classification Tree to the Training set
from sklearn.tree import DecisionTreeClassifier
ct = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0).fit(X_train, y_train)

# Predicting the Test set results
y_pred = ct.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[56  7]
 [ 7 50]]
```

## # 10 Hierarchical Clustering

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Importing the dataset
df = pd.read_csv('bank.csv')
df
```

	<b>id</b>	<b>age</b>	<b>sex</b>	<b>region</b>	<b>income</b>	<b>married</b>	<b>children</b>	<b>car</b>	<b>mortgage</b>	<b>delays</b>
0	ID12101	48	FEMALE	INNER_CITY	17546.00	NO	1	NO	NO	YES
1	ID12102	40	MALE	TOWN	30085.10	YES	3	YES	YES	NO
2	ID12103	51	FEMALE	INNER_CITY	16575.40	YES	0	YES	NO	YES
3	ID12104	23	FEMALE	TOWN	20375.40	YES	3	NO	NO	YES
4	ID12105	57	FEMALE	RURAL	50576.30	YES	0	NO	NO	NO
...	...	...	...	...	...	...	...	...	...	...
595	ID12696	61	FEMALE	INNER_CITY	47025.00	NO	2	YES	YES	NO
596	ID12697	30	FEMALE	INNER_CITY	9672.25	YES	0	YES	NO	YES
597	ID12698	31	FEMALE	TOWN	15976.30	YES	0	YES	NO	YES
598	ID12699	29	MALE	INNER_CITY	14711.80	YES	0	NO	YES	YES
599	ID12700	38	MALE	TOWN	26671.60	NO	0	YES	YES	NO

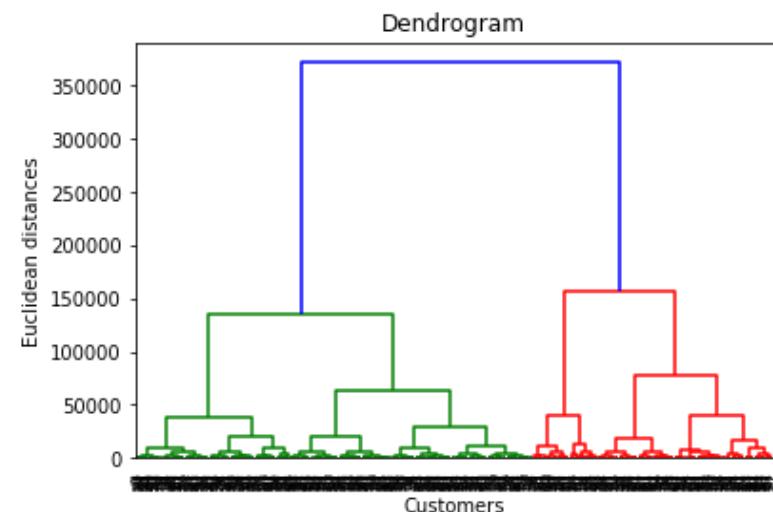
600 rows × 10 columns

```
# Choosing dataset
```

```
X = df.iloc[:, [1, 4]].values
```

```
# Using the dendrogram to find the optimal number of clusters
```

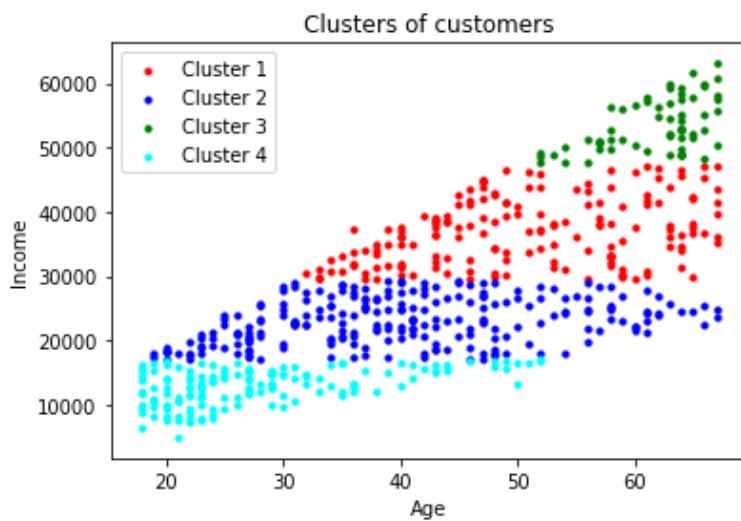
```
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



```
# Fitting Hierarchical Clustering to the dataset
```

```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 4, affinity = 'euclidean', linkage = 'ward').fit_predict(X)
```

```
# Visualising the clusters
plt.scatter(X[hc == 0, 0], X[hc == 0, 1], s = 10, c = 'red',
label = 'Cluster 1')
plt.scatter(X[hc == 1, 0], X[hc == 1, 1], s = 10, c = 'blue',
label = 'Cluster 2')
plt.scatter(X[hc == 2, 0], X[hc == 2, 1], s = 10, c = 'green',
label = 'Cluster 3')
plt.scatter(X[hc == 3, 0], X[hc == 3, 1], s = 10, c = 'cyan',
label = 'Cluster 4')
plt.title('Clusters of customers')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()
```



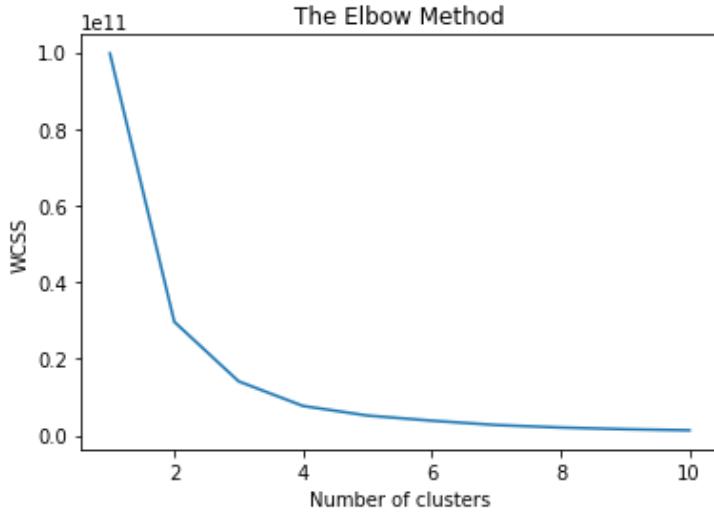
## # 11 K-Means Clustering

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
df = pd.read_csv('bank.csv')

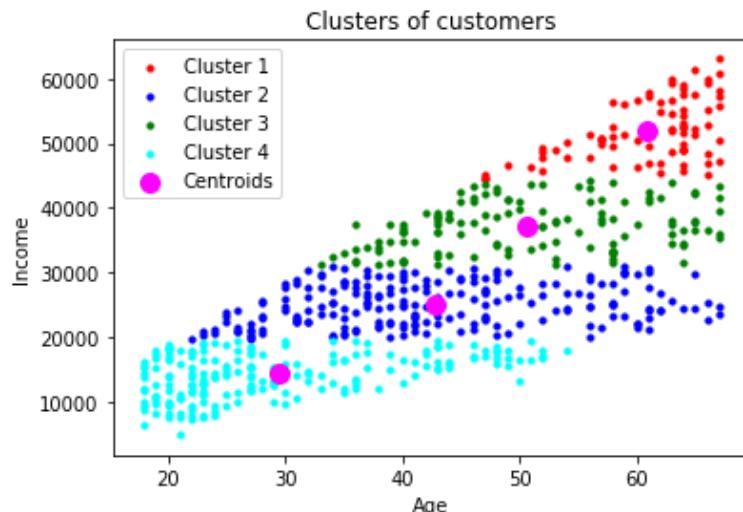
# Choosing dataset
X = df.iloc[:, [1, 4]].values

# Using the elbow method to find the optimal number of
# clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



```
# Fitting K-Means to the dataset
km = KMeans(n_clusters = 4, init = 'k-means++', random_state =
0).fit_predict(X)
kms = KMeans(n_clusters = 4, random_state = 0).fit(X)
kms.cluster_centers_
array([[6.08157895e+01, 5.19730263e+04],
       [4.27536232e+01, 2.51076729e+04],
       [5.06451613e+01, 3.71793032e+04],
       [2.94559585e+01, 1.42846986e+04]])
```

```
# Visualising the clusters
plt.scatter(X[km == 0, 0], X[km == 0, 1], s = 10, c = 'red',
label = 'Cluster 1')
plt.scatter(X[km == 1, 0], X[km == 1, 1], s = 10, c = 'blue',
label = 'Cluster 2')
plt.scatter(X[km == 2, 0], X[km == 2, 1], s = 10, c = 'green',
label = 'Cluster 3')
plt.scatter(X[km == 3, 0], X[km == 3, 1], s = 10, c = 'cyan',
label = 'Cluster 4')
plt.scatter(kms.cluster_centers_[:, 0],
kms.cluster_centers_[:, 1], s = 100, c = 'magenta', label =
'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()
```



## # 12 Natural Language Processing

```
# Importing the libraries
import numpy as np
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Restaurant_Reviews.tsv', delimiter =
'\t', quoting = 3)

# Cleaning the texts
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
corpus = []
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in
set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)

[nltk_data]  Downloading package stopwords to
[nltk_data]      C:\Users\User\AppData\Roaming\nltk_data...
[nltk_data]  Unzipping corpora\stopwords.zip.

# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB().fit(X_train, y_train)

# Predicting the Test set results
y_pred = nb.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[55 42]
 [12 91]]
```

## **# 13 Regression Neural Network**

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
df = pd.read_csv('flats_prep.csv', sep=';')

# Feature Scaling
```

```

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
dfsc = sc.fit_transform(df)
df['rooms'] = dfsc[:,0]
df['m2'] = dfsc[:,3]
df['price'] = dfsc[:,5]

# Cheking correlations
df.corr()

   rooms    location  condition      m2      type     price
rooms  1.000000 -0.060645 -0.342692  0.738398  0.272505  0.594447
location -0.060645  1.000000 -0.122117 -0.288718  0.109806 -0.316291
condition -0.342692 -0.122117  1.000000 -0.082482 -0.769335 -0.163575
m2       0.738398 -0.288718 -0.082482  1.000000 -0.043620  0.885215
type      0.272505  0.109806 -0.769335 -0.043620  1.000000  0.020591
price     0.594447 -0.316291 -0.163575  0.885215  0.020591  1.000000

# Splitting the dataset into the Training set and Test set
X = df.iloc[:, 0:4].values
y = df.iloc[:, 5].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
rnn = Sequential()

# Adding the input layer and the first hidden layer
rnn.add(Dense(output_dim = 6, activation = 'tanh', input_dim =
4))

# Adding the second hidden layer
rnn.add(Dense(output_dim = 6, activation = 'tanh'))

# Adding the output layer
rnn.add(Dense(output_dim = 1, activation = 'linear'))

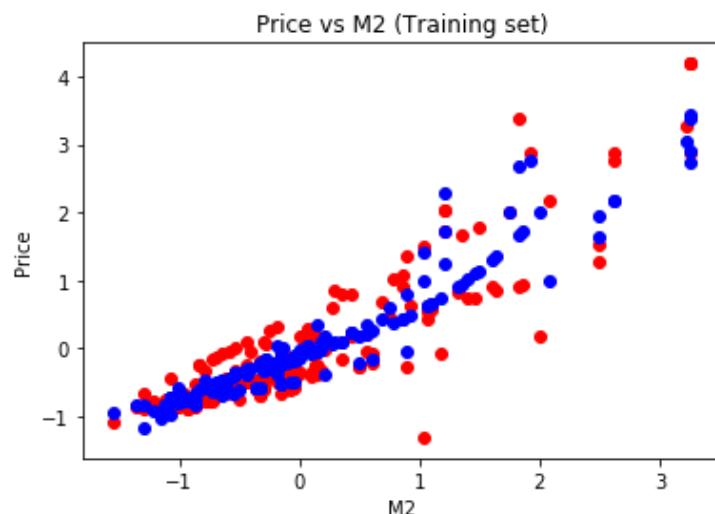
# Compiling the ANN
rnn.compile(optimizer='adam', loss='mean_squared_error',
metrics = ['accuracy'])
loss: 0.1819 - accuracy: 0.0000e+00

# Fitting the ANN to the Training set
rnn.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)

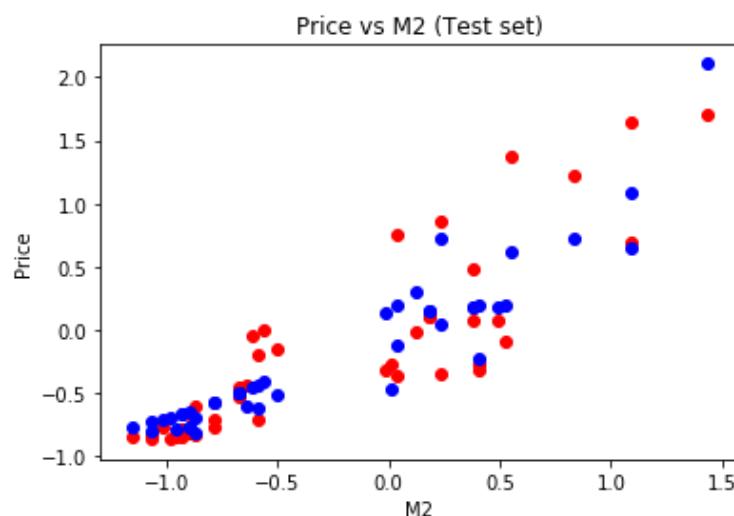
# Predicting the Test set results
y_pred = rnn.predict(X_test)

```

```
# Visualising the Training set results
plt.scatter(X_train[:,3], y_train, color = 'red')
plt.plot(X_train[:,3], rnn.predict(X_train), 'bo')
plt.title('Price vs M2 (Training set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



```
# Visualising the Test set results
plt.scatter(X_test[:,3], y_test, color = 'red')
plt.plot(X_test[:,3], rnn.predict(X_test), 'bo')
plt.title('Price vs M2 (Test set)')
plt.xlabel('M2')
plt.ylabel('Price')
plt.show()
```



## # 14 Classification Neural Network

```

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
df = pd.read_csv('bank_prep.csv')

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
dfsc = sc.fit_transform(df)
df['rooms'] = dfsc[:,0]
df['m2'] = dfsc[:,3]
df['price'] = dfsc[:,5]

# Splitting the dataset into the Training set and Test set (2 variables)
X = df.iloc[:, [0, 2]].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initialising the ANN
scnn = Sequential()

# Adding the input layer and the first hidden layer
scnn.add(Dense(output_dim = 6, init = 'uniform', activation =
'relu', input_dim = 2))

# Adding the second hidden layer
scnn.add(Dense(output_dim = 6, init = 'uniform', activation =
'sigmoid'))

# Adding the output layer
scnn.add(Dense(output_dim = 1, init = 'uniform', activation =
'sigmoid'))

# Compiling the ANN
scnn.compile(optimizer = 'adam', loss = 'binary_crossentropy',
metrics = ['accuracy'])
loss: 0.3049 - accuracy: 0.8521

```

```

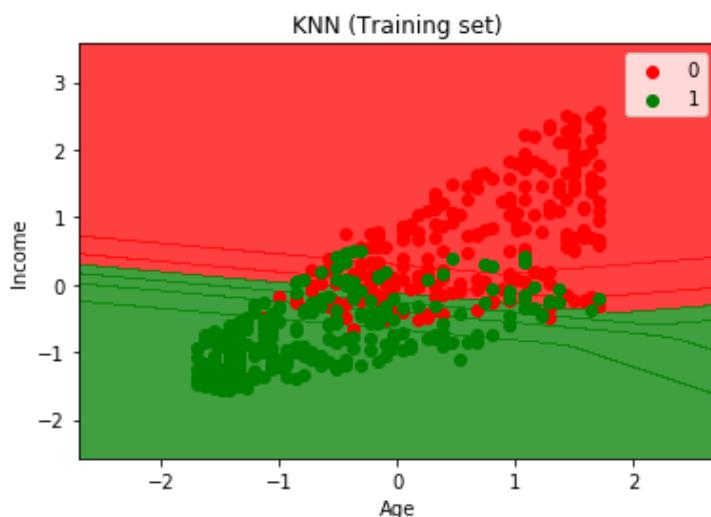
# Fitting the ANN to the Training set
scnn.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)

# Predicting the Test set results
y_pred = scnn.predict(X_test)
y_pred = (y_pred > 0.5)

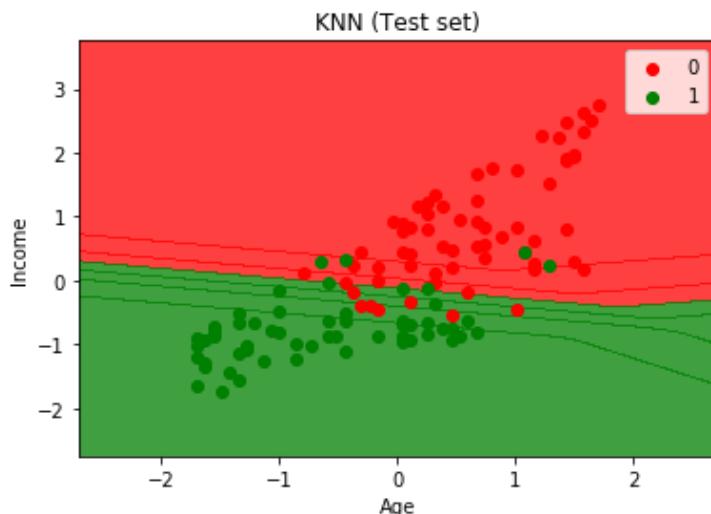
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[56  7]
 [ 7 50]]

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, scnn.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()

```



```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, scnn.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label
= j)
plt.title('NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.legend()
plt.show()
```



```
# Splitting the dataset into the Training set and Test set
X = df.iloc[:, :-1].values
y = df.iloc[:, 7].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

# Initialising the ANN
cnn = Sequential()

# Adding the input layer and the first hidden layer
cnn.add(Dense(output_dim = 6, init = 'uniform', activation =
'relu', input_dim = 10))
```

```
# Adding the second hidden layer
cnn.add(Dense(output_dim = 6, init = 'uniform', activation =
'sigmoid')))

# Adding the output layer
cnn.add(Dense(output_dim = 1, init = 'uniform', activation =
'sigmoid'))

# Compiling the ANN
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy',
metrics = ['accuracy'])
loss: 0.0060 - accuracy: 1.0000

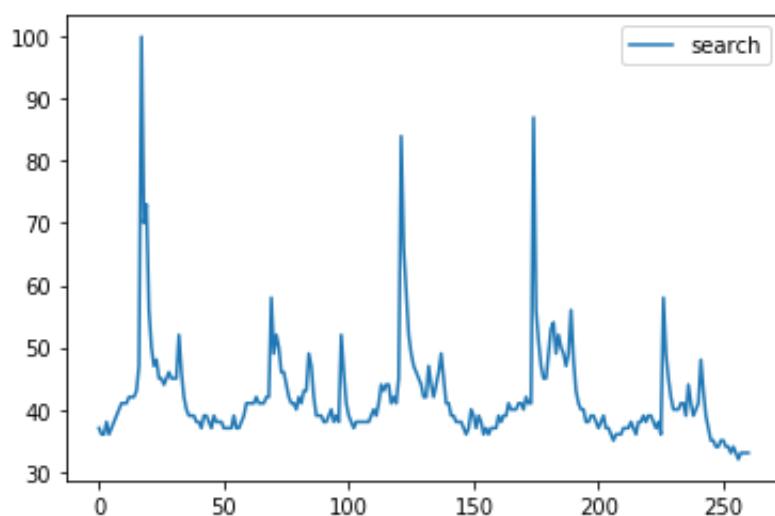
# Predicting the Test set results
y_pred = cnn.predict(X_test)
y_pred = (y_pred > 0.5)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
[[63  0]
 [ 0 57]]
```

## # 15 Forecasting Neural Network

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
ts = pd.read_csv('iPhone.csv')
ts.plot()
```



```

# Checking Stationarity
from statsmodels.tsa.stattools import adfuller
adf = adfuller(ts[['search']])
print('ADF Statistic: %f' % adf[0])
print('p-value: %f' % adf[1])
print('Critical Values:')
for key, value in adf[4].items():
    print('\t%s: %.3f' % (key, value))
# Conclusion: The null hypothesis is rejected, the process
has no unit root, the time series is stationary or does not
have time-dependent structure.
ADF Statistic: -5.668556
p-value: 0.000001
Critical Values:
1%: -3.456
5%: -2.873
10%: -2.573

# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(-1, 1))
ts['sc'] = sc.fit_transform(ts[['search']])

# Creating lagged dataset
ts['x1'] = ts[['sc']].shift(1)
ts['x2'] = ts[['sc']].shift(2)
ts['x3'] = ts[['sc']].shift(3)
ts.columns = ['t', 'search', 'y', 'x1', 'x2', 'x3']
ts

```

	t	search	y	x1	x2	x3
0	2014-05-11	37	-0.852941	NaN	NaN	NaN
1	2014-05-18	36	-0.882353	-0.852941	NaN	NaN
2	2014-05-25	36	-0.882353	-0.882353	-0.852941	NaN
3	2014-06-01	38	-0.823529	-0.882353	-0.882353	-0.852941
4	2014-06-08	36	-0.882353	-0.823529	-0.882353	-0.882353
...	...	...	...	...	...	...
256	2019-04-07	32	-1.000000	-0.970588	-0.941176	-0.970588
257	2019-04-14	33	-0.970588	-1.000000	-0.970588	-0.941176
258	2019-04-21	33	-0.970588	-0.970588	-1.000000	-0.970588
259	2019-04-28	33	-0.970588	-0.970588	-0.970588	-1.000000
260	2019-05-05	33	-0.970588	-0.970588	-0.970588	-0.970588

261 rows × 6 columns

```

# Splitting the dataset into the Training set and Test set
train_size = int(len(ts['y']) * 0.7)
train, test = ts[3:train_size], ts[train_size:]
train_X, train_y = train[['x1', 'x2', 'x3']],
train[['y']].to_numpy()
test_X, test_y = test[['x1', 'x2', 'x3']],
test[['y']].to_numpy()

```

```
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

# Creating model
#np.random.get_state()[1][0]
np.random.seed(605891282)

# Initialising the ANN
fnn = Sequential()

# Adding the input layer and the first hidden layer
fnn.add(Dense(output_dim = 7, activation = 'tanh', input_dim =
3))

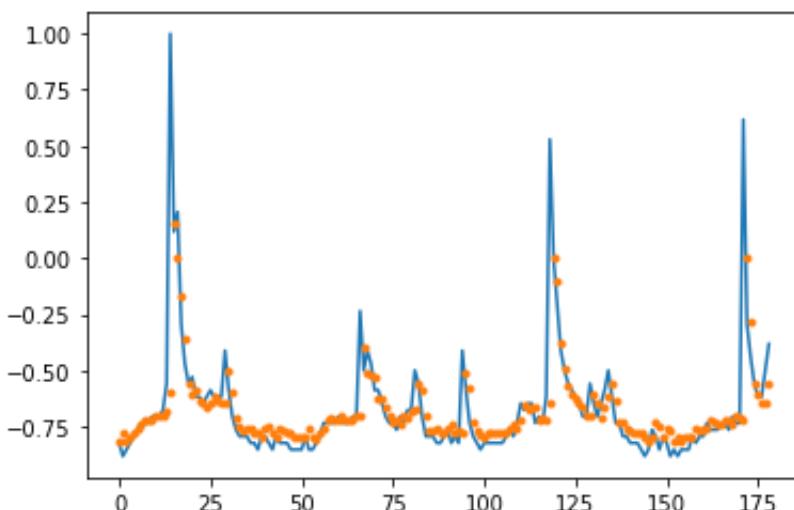
# Adding the second hidden layer
fnn.add(Dense(output_dim = 5, activation = 'linear'))

# Adding the output layer
fnn.add(Dense(output_dim = 1, activation = 'tanh'))

# Compiling the ANN
fnn.compile(optimizer='adam', loss='mean_squared_error',
metrics = ['accuracy'])

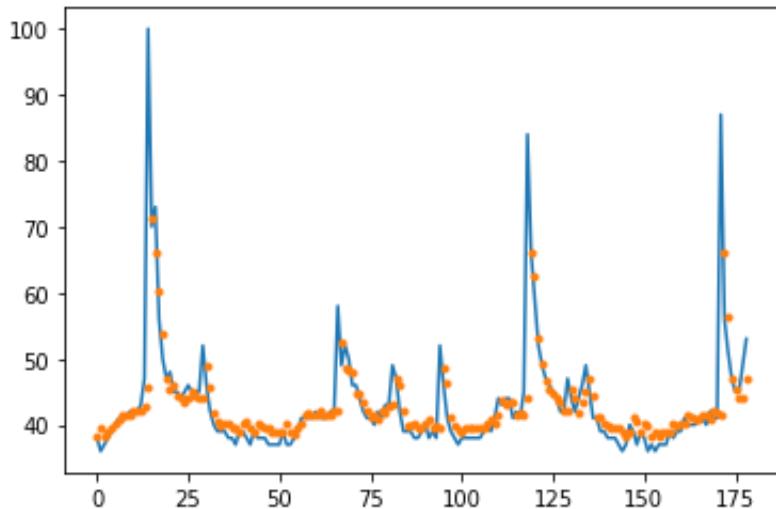
# Fitting the ANN to the Training set
fnn.fit(train_X, train_y, batch_size = 7, nb_epoch = 200)
loss: 0.0390 - accuracy: 0.0056

# Predicting scaled training set
yhat = fnn.predict(train_X)
from matplotlib import pyplot
pyplot.plot(train_y)
pyplot.plot(yhat, '.')
pyplot.show()
```



**# Inversing scaling**

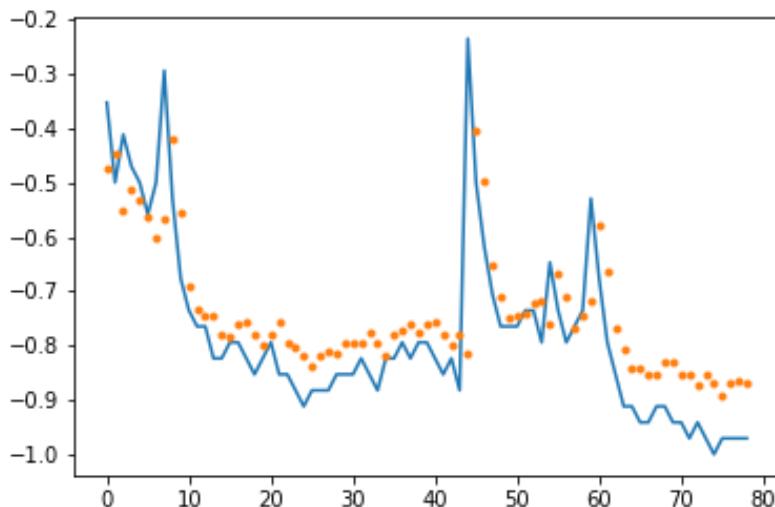
```
tshat = pd.DataFrame(yhat, columns=['y'])
from sklearn.preprocessing import MinMaxScaler
tshat['searchhat'] = sc.inverse_transform(tshat)
tshat['search'] = train['search'][0:][].reset_index()['search']
pyplot.plot(tshat['search'])
pyplot.plot(tshat['searchhat'], '.')
pyplot.show()
```

**# Reporting performance**

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(tshat['search'],
tshat['searchhat']))
print('Test RMSE: %.3f' % rmse)
Test RMSE: 6.660
```

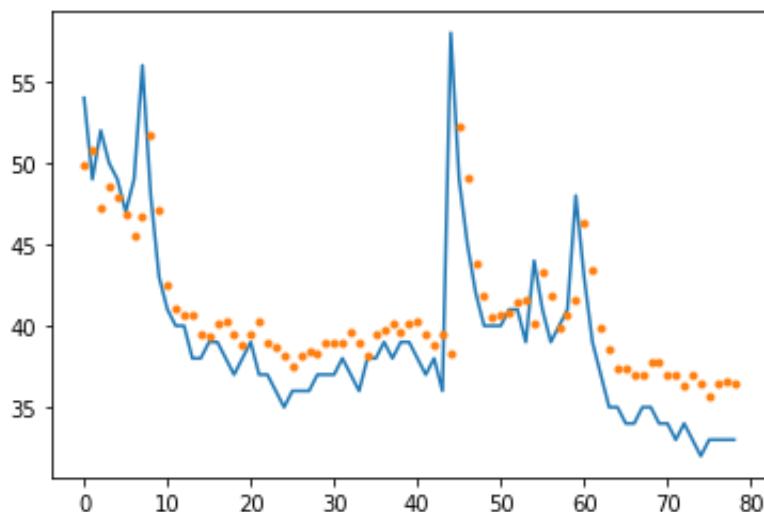
**# Predicting scaled test set**

```
X1, y1 = test_X, test_y
yhat1 = fnn.predict(X1)
from matplotlib import pyplot
pyplot.plot(y1)
pyplot.plot(yhat1, '.')
pyplot.show()
```



**# Inversing scaling**

```
tshat1 = pd.DataFrame(yhat1, columns=['y'])
from sklearn.preprocessing import MinMaxScaler
tshat1['searchhat'] = sc.inverse_transform(tshat1)
tshat1['search'] = test['search'].reset_index()['search']
pyplot.plot(tshat1['search'])
pyplot.plot(tshat1['searchhat'], '.')
pyplot.show()
```



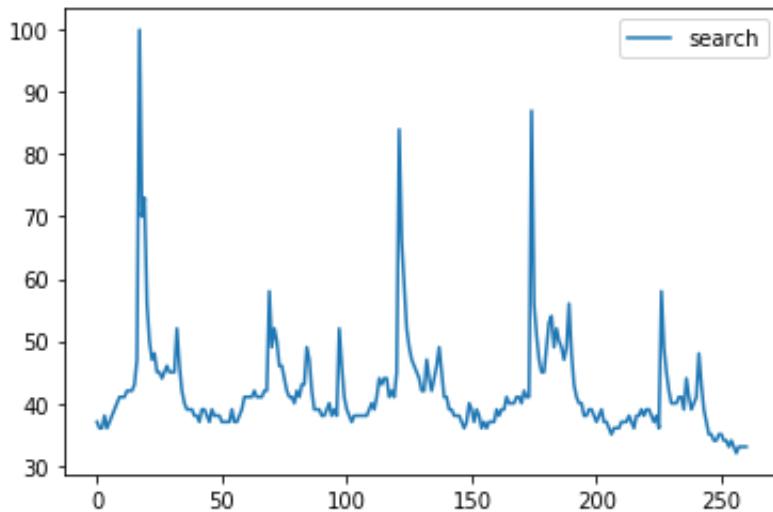
**# Reporting performance**

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(tshat1['search'],
tshat1['searchhat']))
print('Test RMSE: %.3f' % rmse)
Test RMSE: 4.016
```

## # 16 LSTM

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
ts = pd.read_csv('iPhone.csv')
ts.plot()
```



## # Checking Stationarity

```
from statsmodels.tsa.stattools import adfuller
adf = adfuller(ts[['search']])
print('ADF Statistic: %f' % adf[0])
print('p-value: %f' % adf[1])
print('Critical Values:')
for key, value in adf[4].items():
    print('\t%s: %.3f' % (key, value))
# Conclusion: The null hypothesis is rejected, the process
has no unit root, the time series is stationary or does not
have time-dependent structure.
ADF Statistic: -5.668556
p-value: 0.000001
Critical Values:
    1%: -3.456
    5%: -2.873
    10%: -2.573
```

## # Feature Scaling

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(-1, 1))
ts['sc'] = sc.fit_transform(ts[['search']])
```

## # Creating lagged dataset

```
ts['x'] = ts[['sc']].shift(1)
```

```
ts.columns = ['t','search','y','x']

# Splitting the dataset into the Training set and Test set
train_size = int(len(ts[['x']]) * 0.7)
train, test = ts[1:train_size], ts[train_size:]
train_X, train_y = train[['x']].to_numpy(),
train[['y']].to_numpy()
test_X, test_y = test[['x']].to_numpy(),
test[['y']].to_numpy()

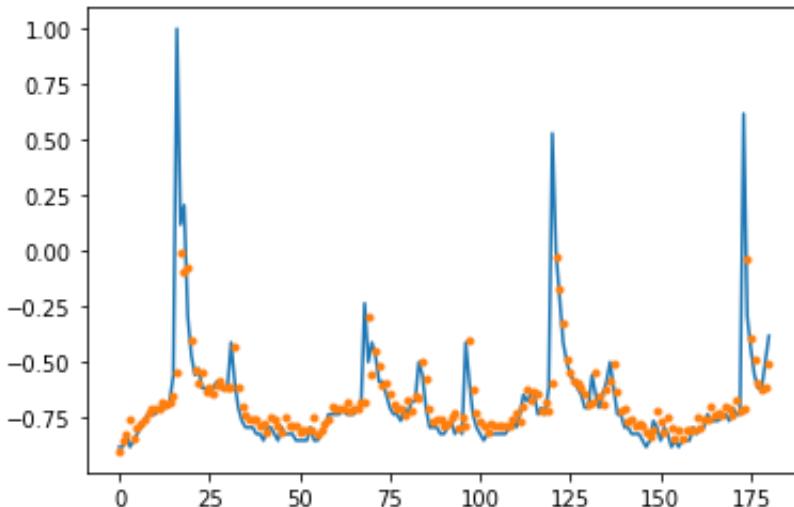
# Importing the Keras libraries and packages
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
#np.random.seed(2147483648)

# Creating model
X, y = train_X, train_y
X = X.reshape(X.shape[0], 1, X.shape[1])
batch_size = 1
nb_epoch = 100
neurons = 5
lstm = Sequential()
lstm.add(LSTM(neurons, batch_input_shape=(batch_size,
X.shape[1], X.shape[2]), stateful=True))
lstm.add(Dense(1))

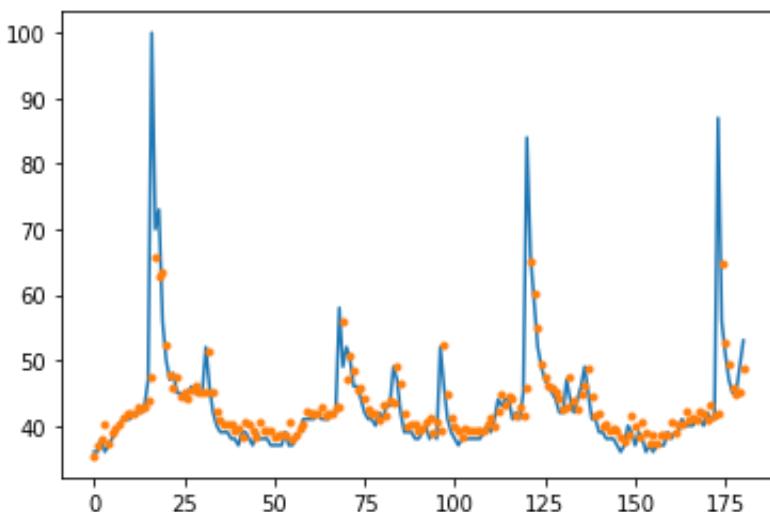
# Compiling
lstm.compile(loss='mean_squared_error', optimizer='adam')

# Fitting
for i in range(nb_epoch):
    lstm.fit(X, y, epochs=1, batch_size=batch_size, verbose=0,
shuffle=False)
    lstm.reset_states()

# Predicting scaled training set
yhat = lstm.predict(X, batch_size=batch_size)
from matplotlib import pyplot
pyplot.plot(y)
pyplot.plot(yhat, '.')
pyplot.show()
```

**# Inversing scaling**

```
tshat = pd.DataFrame(yhat, columns=['y'])
from sklearn.preprocessing import MinMaxScaler
tshat['searchhat'] = sc.inverse_transform(tshat)
tshat['search'] = train['search'][0:].reset_index()['search']
pyplot.plot(tshat['search'])
pyplot.plot(tshat['searchhat'], '.')
pyplot.show()
```

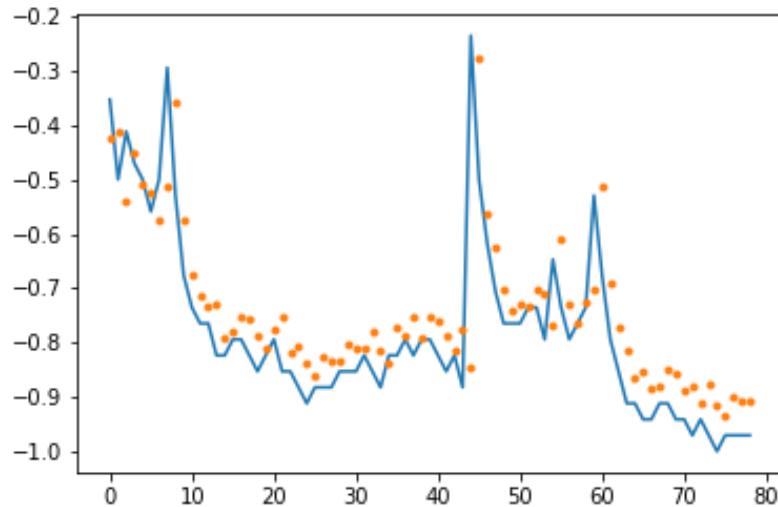
**# Reporting performance**

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(tshat['search'],
tshat['searchhat']))
print('Test RMSE: %.3f' % rmse)
Test RMSE: 6.515
```

**# Predicting scaled test set**

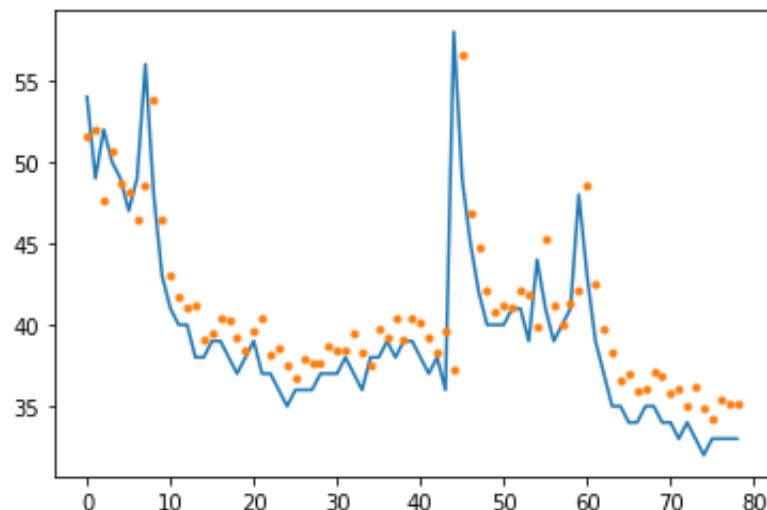
```
X1, y1 = test_X, test_y
X1 = X1.reshape(X1.shape[0], 1, X1.shape[1])
yhat1 = lstm.predict(X1, batch_size=batch_size)
```

```
from matplotlib import pyplot
pyplot.plot(y1)
pyplot.plot(yhat1, '.')
pyplot.show()
```



#### # Inversing scaling

```
tshat1 = pd.DataFrame(yhat1, columns=['y'])
from sklearn.preprocessing import MinMaxScaler
tshat1['searchhat'] = sc.inverse_transform(tshat1)
tshat1['search'] = test['search'].reset_index()['search']
pyplot.plot(tshat1['search'])
pyplot.plot(tshat1['searchhat'], '.')
pyplot.show()
```



#### # Reporting performance

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(tshat1['search'],
tshat1['searchhat']))
print('Test RMSE: %.3f' % rmse)
Test RMSE: 3.533
```

## # 17 Convolutional Neural Network

```

from __future__ import print_function

# Importing the Keras libraries and packages
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

# Preparing data
batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows,
img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows,
img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows,
img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows,
img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
Downloading data from https://s3.amazonaws.com/img-
datasets/mnist.npz
11493376/11490434 [=====] - 2s
0us/step
x_train shape: (60000, 28, 28, 1)

```

## Кононова К. Ю.

```
60000 train samples
10000 test samples
```

```
# Creating and fitting model
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

Train on 60000 samples, validate on 10000 samples
Test loss: 0.028183125485236268
Test accuracy: 0.991100013256073
```

## ЛІТЕРАТУРА

### Основна

1. Айвазян С. А. Прикладная статистика: классификация и снижение размерности / С. А. Айвазян, В. М. Бухштабер, И. С. Енюков, Л. Д. Мешалкин. М.: Финансы и статистика, 1989. 607 с.
2. Айвазян С. А. Прикладная статистика и основы эконометрии / С. А. Айвазян, В. С. Мхитарян. М.: ЮНИТИ, 1998. 1022 с.
3. Арапов М. В. О смысле ранговых распределений / М. В. Арапов, Е. Н. Ефимова, Ю. А. Шрейдер // Науч.-техн. информ. 1975. Сер. 2. № 1. С. 9–20.
4. Афифи А. Статистический анализ: подход с использованием ЭВМ / А. Афифи, С. Эйзен. М.: Мир, 1982. 488 с.
5. Барсегян А. А. Технологии анализа данных. Data Mining, Visual Mining, Text Mining, OLAP / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод. БХВ-Петербург, 2007.
6. Барсегян А. А. Анализ данных и процессов / А. А. Барсегян, М. С. Куприянов, И. И. Холод. СПб.: БХВ-Петербург, 2009. 512 с.
7. Бирюкова С. Анализ последовательностей в R: TraMineR. НИУ ВШЭ. 2014. URL: <https://www.hse.ru>.
8. Бокс Дж. Анализ временных рядов. Прогноз и управление / Дж. Бокс, Г. Дженкинс. М.: Мир, 1974. Вып. 1. 406 с.
9. Бонгард М. М. Проблема узнавания. М.: Наука, 1967. 320 с.
10. Вайнцвайг М. Н. Алгоритм обучения распознаванию образов “Кора” // Алгоритмы обучения распознаванию образов. Ред. В. Н. Вапник. М.: Сов. радио, 1973. С. 110–116.
11. Вапник В. Н. Алгоритмы и программы восстановления зависимостей. М.: Наука, 1984. 816 с.

12. Вапник В. Н. Теория распознавания образов / В. Н. Вапник, А. Я. Червоненкис. М.: Наука, 1974. 487 с.
13. Венэблз У. Н. Введение в R. Заметки по R: среда программирования для анализа данных и графики / У. Н. Венэблз, Д. М. Смит. Москва, 2014. 109 с.
14. Горбань А. Н. Нейроинформатика / А. Н. Горбань, В. Л. Дунин-Барковский, Е. М. Миркес. Новосибирск: Наука. Сиб. предприятие РАН, 1998. 296 с.
15. Горбач А. Н. Покупательское поведение: анализ спонтанных последовательностей и регрессионных моделей в маркетинговых исследованиях / А. Н. Горбач, Н. А. Цейтлин. Киев: Освіта України, 2011. 220 с.
16. Гудфеллоу Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль. М.: ДМК Пресс, 2018.
17. Джеймс Г. Введение в статистическое обучение с примерами на языке R / Г. Джеймс, Д. Уиттон, Т. Хасти, Р. Тибштадт. М.: ДМК Пресс, 2016. 450 с.
18. Джулли А. Библиотека Keras – инструмент глубокого обучения / А. Джулли, С. Пал. ДМК Пресс, 2018.
19. Дрейпер Н. Прикладной регрессионный анализ / Н. Дрейпер, Г. Смит. М.: Финансы и статистика. Кн. 1, 1986. 366 с. Кн. 2, 1987. 352 с.
20. Дэйвисон М. Многомерное шкалирование. Методы наглядного представления данных. М.: Финансы и статистика, 1988. 348 с.
21. Дюк В. А. Data Mining: учебный курс / В. А. Дюк, А. П. Самойленко. СПб.: Питер, 2001. 368 с.
22. Загоруйко Н. Г. Прикладные методы анализа данных и знаний. Новосибирск: ИМ СО РАН, 1999. 270 с.
23. Зайцев К. С. Применение методов Data Mining для поддержки процессов управления ИТ-услугами. М.: МИФИ, 2009. 96 с.

24. Зарядов И. С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика. Москва: Изд-во РУДНБ, 2010. 207 с.
25. Зарядов И. С. Статистический пакет R: теория вероятностей и математическая статистика. Москва: Изд-во РУДНБ, 2010. 141 с.
26. Зиновьев А. Ю. Визуализация многомерных данных. Красноярск: КГТУ, 2000. 168 с.
27. Кабаков Р. И. R в действии: анализ и визуализация данных в программе. М.: ДМК Пресс, 2014. 580 с.
28. Кендалл М. Многомерный статистический анализ и временные ряды / М. Кендалл, А. Стьюарт. М.: Наука, 1976. 736 с.
29. Кендалл М. Статистические выводы и связи/ М. Кендалл, А. Стьюарт. М.: Наука, 1973. 899 с.
30. Ким Дж.-О. Факторный, дискриминантный и кластерный анализ / Дж.- О. Ким, Ч. У. Мюллер, У. Р. Клекка. М.: Финансы и статистика, 1989. 215 с.
31. Классификация и кластер / под ред. Дж. Вэн-Райзина. М.: Мир, 1980. 390 с.
32. Кобзарь А. И. Прикладная математическая статистика. Для инженеров и научных работников. М.: Физматлит, 2006. 816 с.
33. Краскэл Дж. Б. Многомерное шкалирование и другие методы поиска структуры // Статистические методы для ЭВМ / под ред. К. Энслейна, Э. Рэлстона, Г. С. Уилфа. М.: Наука, 1986. С. 301–347.
34. Криват Б. Microsoft SQL Server 2008: Data Mining – интеллектуальный анализ данных / Б. Криват, Д. Макленнен, Ч. Танг. ВНВ, 2009.
35. Лбов Г. С. Методы обработки разнотипных экспериментальных данных. Новосибирск: Наука, 1981. 160 с.

36. Ллойд Э. Справочник по прикладной статистике. В 2-х т. / Э. Ллойд, У. Ледерман. М.: Финансы и статистика. Т. 1. 1989. 510 с., Т. 2. 1990. 526 с.
37. Люк Д. Анализ сетей (графов) в среде R. Руководство пользователя. М.: ДМК Пресс, 2016.
38. Мастицкий С. Э. Визуализация данных с помощью ggplot2. М.: ДМК Пресс, 2016.
39. Мастицкий С. Э. Статистический анализ и визуализация данных с помощью R / С. Э. Мастицкий, В. К. Шитиков. М.: ДМК Пресс, 2015. 496 с. URL: <http://ievbras.ru/ecostat>.
40. Налимов В. В. Теория эксперимента. М.: Наука, 1971. 207 с.
41. Огнева Д. Пакет “arules” системы R. МГУ им. Ломоносова. 2012. URL: [www.machinelearning.ru](http://www.machinelearning.ru).
42. Паклин Н. Б. Бизнес-аналитика: от данных к знаниям / Н. Б. Паклин, В. И. Орешков. СПб.: Изд. Питер, 2009.
43. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. М.: ДМК Пресс, 2015. 400 с.
44. Форрестер Дж. Антиинтуитивное поведение сложных систем // Современные проблемы кибернетики. М.: Знание, 1977. С. 9–25.
45. Хокинс Д. Об интеллекте / Д. Хокинс, С. Блейксли. М.: ООО «ИД Вильямс», 2007. 204 с.
46. Храмов Д. А. Сбор данных в Интернете на языке R. М.: ДМК Пресс, 2016.
47. Черняк А. И. Интеллектуальный анализ данных: учебник / А. И. Черняк, П. В. Захарченко; Киевский национальный университет імені Т. Шевченко. К.: Знание, 2014. 599 с.

48. Чубукова И. А. Data Mining: учебное пособие. – М.: Интернет-университет информационных технологий: БИНОМ: Лаборатория знаний, 2006.
49. Шипунов А. Б. Наглядная статистика. Используем R! М.: ДМК Пресс, 2014. 298 с.
50. Шитиков В. К. Рандомизация и бутстреп: статистический анализ в биологии и экологии с использованием R / В. К. Шитиков, Г. С. Розенберг. Тольятти : Кассандра, 2014. 314 с. URL: <http://ievbras.ru/ecostat>.
51. Эфрон Б. Нетрадиционные методы многомерного статистического анализа. М.: Финансы и статистика, 1988. 263 с.

### Допоміжна література

52. Agrawal R., Imielinski T., Swami A. (1993). Mining Associations between Sets of Items in Massive Databases. In Proc. of the 1993 ACM-SIGMOD Int'l Conf. on Management of Data, 207–216.
53. Agrawal R., Srikant R. (1994). "Fast Discovery of Association Rules", In Proc. of the 20th International Conference on VLDB, Santiago, Chile, September 1994.
54. Bartholomew, D. J. (1987). Latent variable models and factor analysis. Oxford University Press.
55. Basilevsky, A. (1994). Statistical Factor Analysis and Related Methods: Theory and Applications. Wiley.
56. Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. In Large Scale Kernel Machines.
57. Bertsekas, D. P. and Tsitsiklis, J. (1996). Neuro-Dynamic Programming. Athena Scientific.
58. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

59. Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning in Neural Networks*. Cambridge University Press, Cambridge, UK.
60. Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.
61. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA.
62. Brett Lantz. *Machine Learning with R*. Packt Publishing, Birmingham – Mumbai, 2013.
63. Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning*. MIT Press, Cambridge, MA.
64. Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2014). The loss surface of multilayer networks.
65. Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*, 2nd Edition. Wiley-Interscience.
66. David J. C. MacKay (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.
67. Devroye, L. (2013). *Non-Uniform Random Variate Generation*. SpringerLink: Bücher. Springer New York.
68. Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., and Talay, S. (2011). Large-scale FPGA-based convolutional networks. In R. Bekkerman, M. Bilenko, and J. Langford, editors, *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press.
69. Frey, B. J. (1998). *Graphical models for machine learning and digital communication*. MIT Press.
70. Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer.

71. Hand D., Manila H., and Smyth P. (2001) Principles of Data Mining, MIT Press.
72. Hastie, T., Tibshirani, R., and Friedman, J. (2001). The elements of statistical learning: data mining, inference and prediction. Springer Series in Statistics. Springer Verlag.
73. Hebb, D. O. (1949). The Organization of Behavior. Wiley, New York.
74. Hinton, G. E. and Sejnowski, T. J. (1999). Unsupervised learning: foundations of neural computation. MIT press.
75. Jaynes, E. T. (2003). Probability Theory: The Logic of Science. Cambridge University Press.
76. Kohonen, T., Honkela, T. (2007). Kohonen Network. Scholarpedia.
77. Koller, D. and Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. MIT Press.
78. Lenat, D. B. and Guha, R. V. (1989). Building large knowledge-based systems; representation and inference in the Cyc project. Addison-Wesley Longman Publishing Co., Inc.
79. Luenberger, D. G. (1984). Linear and Nonlinear Programming. Addison Wesley.
80. Machine Learning A-Z™: Hands-On Python & R In Data Science. <https://www.udemy.com/course/machinelearning>.
81. MacKay, D. (2003). Information Theory, Inference and Learning Algorithms. Cambridge University Press.
82. Mills, T.C. (1990). Time Series Techniques for Economists. Cambridge University Press.
83. Minsky, M. L. and Papert, S. A. (1969). Perceptrons. MIT Press, Cambridge
84. Mitchell, T. M. (1997). Machine Learning. McGraw-Hill, New York.
85. Murphy, K. P. (2012). Machine Learning: a Probabilistic Perspective. MIT Press, Cambridge, MA, USA.
86. Nocedal, J. and Wright, S. (2006). Numerical Optimization. Springer.

87. Roberts, S. and Everson, R. (2001). Independent component analysis: principles and practice. Cambridge University Press.
88. Rokach, L., Oded, M. (2005) Clustering methods. Data mining and knowledge discovery handbook. Springer US.
89. Rosenblatt, F. (1962). Principles of Neurodynamics. Spartan, New York.
90. Russel, S. J. and Norvig, P. (2003). Artificial Intelligence: a Modern Approach. Prentice Hall.
91. Sutton, R. and Barto, A. (1998). Reinforcement Learning: An Introduction. MIT Press.
92. Tan, P., Steinbach, M., Kumar, V. (2005). Introduction to Data Mining, Addison Wesley.
93. Vapnik, V. N. (1982). Estimation of Dependences Based on Empirical Data. Springer-Verlag, Berlin.
94. Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. Springer, New York.
95. Ville B. (2001). Microsoft Data Mining. Digital Press.
96. Witten I. H., Frank E. (2005). Data Mining: Practical machine learning tools and techniques, 2<sup>nd</sup> Edition, Morgan Kaufmann, San Francisco.

**Посилання на інформаційні ресурси в Інтернеті, відео-лекції, інше методичне забезпечення**

97. <http://alexanderdyakonov.narod.ru/upR.pdf>.
98. <http://algorithmist.ru/2011/05/clustering-with-example-in-r.html>.
99. <http://cran.gis-lab.info/web/packages/ROCR/ROCR.pdf>.
100. <http://mymagictools.blogspot.com/2015/07/r.html?view=classic>.
101. <http://package.rdatamining.com>.
102. <http://qaru.site/questions/19913/cluster-analysis-in-r-determine-the-optimal-number-of-clusters>.
103. <http://rdatamining.com/docs/r-reference-card-for-data-mining>.

104. [http://re9ulus.github.io/2015/12/07/trees\\_in\\_r](http://re9ulus.github.io/2015/12/07/trees_in_r).
105. <http://rocr.bioinf.mpi-sb.mpg.de>.
106. <http://rwanjohi.rbind.io/2018/04/05/time-series-forecasting-using-lstm-in-r>.
107. <https://aakinshin.net/ru/posts/r-functions>.
108. [https://clarkdatalabs.github.io/soms/SOM\\_NBA](https://clarkdatalabs.github.io/soms/SOM_NBA).
109. <https://cran.r-project.org/doc/contrib/Short-refcard.pdf>.
110. <https://cran.r-project.org/web/packages>.
111. <https://datacamp.com/community/tutorials/support-vector-machines-r>.
112. <https://habr.com/ru/company/wunderfund/blog/331310>.
113. <https://habr.com/ru/post/105220>.
114. <https://habrahabr.ru/company/lanit/blog/328858>.
115. <https://habrahabr.ru/company/ods/blog/322534>.
116. <https://habrahabr.ru/company/ods/blog/323890>.
117. <https://habrahabr.ru/company/ods/blog/324402>.
118. <https://habrahabr.ru/company/ods/blog/325422>.
119. <https://habrahabr.ru/company/ods/blog/325654>.
120. <https://habrahabr.ru/company/ods/blog/327242>.
121. <https://ranalytics.github.io/data-mining>.
122. <https://r-statistics.co/ggplot2-cheatsheet.html#Hide%20legend>.
123. <https://shanelynn.ie/self-organising-maps-for-customer-segmentation-using-r>.

## Матеріали курсу

<https://github.com/katerynakononova/ml>

Кононова К. Ю.

Навчальне видання

**Кононова** Катерина Юріївна

**Машинне навчання: методи та моделі**

Коректор Б. О. Хільська  
Комп'ютерне верстання  
Макет обкладинки

Формат \_\_\_\_\_. Ум. друк. арк. \_\_\_\_\_. Наклад \_\_\_\_\_. пр. Зам. № 200/19.

Видавець і виготовлювач –  
Харківський національний університет імені В. Н. Каразіна,  
61002, м. Харків, майдан Свободи, 4.  
Свідотство суб'єкта видавничої справи ДК № 3367 від 13.01.2009

Видавництво ХНУ імені В. Н. Каразіна  
Тел. 705-24-32