СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

імені ВОЛОДИМИРА ДАЛЯ

(м. Сєвєродонецьк)

Факультет Інформаційних технологій та електроніки
(повне найменування факультету)

Кафедра Програмування та математики
(повна назва кафедри)

# ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

освітньо-кваліфікаційного рівня магістр
(бакалавр, спеціаліст, магістр)

напряму підготовки 122 Комп'ютерні науки та інформаційні технології
(шифр і назва напряму підготовки)

спеціальності Інформатика
(шифр і назва спеціальності)

на тему "Рекомендаційна система для інтернет-магазина"

Виконав: студент групи ІНФ-16дм

Шапошнікова Є.А. …….

(прізвище, та ініціали) (підпис)

Керівник Фесенко Т.М. …….

(прізвище та ініціали) (підпис)

Завідувач кафедри Лифар В.О. …….

(прізвище та ініціали) (підпис)

Рецензент . …….

(прізвище та ініціали) (підпис)

Сєвєродонецьк - 2018

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

імені ВОЛОДИМИРА ДАЛЯ

(м. Сєвєродонецьк)

Факультет Інформаційних технологій та електроніки

Кафедра Програмування та математики
Освітньо-кваліфікаційний рівень магістр

(бакалавр, спеціаліст, магістр)

Спеціальність Інформатика

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____ В.О.Лифар
“____” _____ 2017 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Шапошнікової Євгенії Анатоліївни

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Рекомендаційна система для інтернет-магазина

керівник проекту (роботи) к.т.н., доц. Фесенко Т.М.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “08”листопада 2017 року

2. Строк подання студентом проекту (роботи) 15.01.2018

3. Вихідні дані до проекту (роботи) Матеріали науково-дослідної практики

_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Актуальність теми і постановка завдань дослідження.

2) Аналіз напрямку дослідження;

3) Огляд та вибір методів розробки програми;

4) Розробка програми для інтелектуального аналізу даних у вигляді.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання_____08.11.2017_____

**КАЛЕНДАРНИЙ ПЛАН**

| № з/п | Назва етапів дипломного проекту (роботи) | Строк виконання етапів проекту ( роботи ) | Примітка |
|---|---|---|---|
| 1 | Актуальність теми і постановка завдань дослідження. | 08.11.17-10.11.17 | |
| 2 | Аналіз розробок та досліджень | 11.11.17-12.11.17 | |
| 3 | Аналіз методів кластерізації даних | 13.11.17-01.12.17 | |
| 4 | Огляд та вибір інструментів розробки програми | 02.12.17-30.12.17 | |
| 5 | Аналіз алгоритму та програмна реалізація системи | 31.12.17-04.01.18 | |
| 6 | Оформлення пояснювальної записки | 05.01.18-15.01.18 | |

Студент _____ Шапошнікова Є.А.

( підпис ) (прізвище та ініціали)

Керівник проекту (роботи) _____ ____Фесенко Т.М_____

( підпис ) (прізвище та ініціали)

# ЗМІСТ

# ВСТУП

Повсюдне використання комп'ютерів призвело до розуміння важливості завдань, пов'язаних з аналізом накопиченої інформації з метою отримання нових знань. Виникла потреба у створенні сховищ даних і систем підтримки прийняття рішень, заснованих, зокрема, на методах теорії штучного інтелекту.

Дійсно, управління підприємством, банком, різні сфери бізнесу, в тому числі електронного, немислимі без процесів накопичення, аналізу, виявлення певних закономірностей і залежностей, прогнозування тенденцій і ризиків.

Давній інтерес до методів, алгоритмічним моделей та засобів їх реалізації, використовуються на етапі аналізу даних, з'явився причиною розвитку даної теми.

Дуже мала частина даних буде колись побачена неозброєним людським оком через її величезну кількість. Тому треба знаходити корисну інформацію в цьому океані знань за допомогою застосування методів Data Mining.

Алгоритми, що використовуються в Data Mining, вимагають великої кількості обчислень. Раніше це було стримуючим фактором широкого практичного застосування Data Mining, проте сьогоднішнє зростання продуктивності сучасних процесорів зняв гостроту цієї проблеми. Тепер за прийнятний час можна провести якісний аналіз сотень тисяч і мільйонів записів.

# РОЗДІЛ 1. ВИБІР НАПРЯМКУ ДОСЛІДЖЕННЯ

## 1.1 Аналіз даних

Корпорація EMC оприлюднила результати дослідження Big Data, Bigger Digital Shadows and Biggest Growth in the Far East, проведеного IDC за підтримки компанії EMC. Це дослідження демонструє безпрецедентне зростання інформації в світі, тільки 0,4% якої, за оцінками IDC, аналізується [1].

Повсюдне поширення технологій і доступу до Інтернету привели до подвоєння обсягу інформації за останні роки. Дослідження оцінило обсяг згенерованих даних в 2012 р в 2,8 зеттабайт і прогнозує до 2020 р збільшення обсягу до 40 зеттабайт, що перевершує попередні прогнози на 14% [1].

Лише незначна частка потенціалу «великих даних» (Big Data) використовується у всьому світі, незважаючи на те, що обсяги корисної інформації продовжують рости.

За прогнозами IDC, до 2020 р цифрова всесвіт досягне обсягу в 40 зеттабайт, що перевершує попередній прогноз на 5 зеттабайт. Всього з початку 2010 р обсяг даних зріс в 50 разів. За даними дослідження, по всьому світу буде створено і використано 2,8 зеттабайт даних [1].

Зростання загального обсягу інформації відбувається в основному за рахунок автоматично генеруються даних - до 2020 року їх обсяг повинен збільшитися в 15 разів.

До 2020 р ринки, що розвиваються стануть основним постачальником інформації.

За прогнозами, інвестиції в IT-інфраструктуру цифрового всесвіту (зберігання і управління інформацією, обладнання, телекомунікації та персонал) в період з 2012 по 2020 р виростуть на 40%. Інвестиції в збереження та захист інформації, Big Data і Cloud Computing будуть рости значно швидше.

Ринки, що розвиваються будуть генерувати все більшу частку інформації в загальному обсязі: в 2010 р їх частка була 23% цифрового всесвіту, але вже до 2012 р зросла до 36%. За прогнозами IDC, до 2020 р 62% даних буде пов'язано з ринками, що розвиваються [1].

Потреба в аналізі даних вийшла далеко за межі технологічних і інтернет-компаній. Методи машинного навчання все активніше використовуються в абсолютно різних областях, аж до оптимізації маршрутів транспорту. З їх допомогою створюються нові ліки і автомобілі без водія, підбирається музика під настрій, знаходяться потенційні супутники життя.

Кількість даних зростає зі швидкістю, яку людині неможливо уявити. Дані ніщо без аналізу. Тільки неймовірна кількість закодованої в одиниці і нулі інформації. Тому будують нові дата-центри, а в їх глибинах зберігається, а також обробляє безліч даних.

Ми всі чули про контекстну рекламу, показ якої грунтується на наших перевагах, про які пошукові машини дізнаються з наших дій онлайн. Але ось про інші сфери мало хто говорить широкому загалу. Але ж крім того, що біг дата в сумі з прогнозної аналітикою дозволяє рекламодавцям і банкам заробляти неймовірні гроші, вони допомагають рятувати людські життя.

Область Data Mining виросла з одного семінару проведеного Григорієм Пятецкий-Шапіро в 1989 році до десятків міжнародних конференцій в 2003 році з тисячами дослідників у багатьох країнах світу. Раніше, працюючи в компанії GTE Labs, Григорій Пятецкий-Шапіро зацікавився питанням: чи можна автоматично знаходити певні правила, щоб прискорити деякі запити до великих баз даних. Тоді ж було запропоновано два терміни - data mining ( «видобуток даних») і knowledge discovery in data (який слід перекладати як «відкриття знань в базах даних»).

У 1975 році з'явився перший стандарт асоціації по мовам систем обробки даних - Conference on Data System Languages (CODASYL), який визначив ряд фундаментальних понять в теорії систем баз даних, які до сих пір є основоположними для мережевої моделі даних. У подальший розвиток

теорії баз даних великий внесок був зроблений американським математиком Е.Ф. Коддом, який є творцем реляційної моделі даних.

Протягом цього періоду багато дослідників експериментували з новим підходом в напрямках структуризації баз даних і забезпечення доступу до них. Метою цих пошуків було отримання реляційних прототипів для більш простого моделювання даних. В результаті, в 1985 році була створена мова, названий SQL. На сьогоднішній день практично всі СУБД забезпечують даний інтерфейс.

З'явилися специфічні типи даних – «графічний образ», «документ», «звук», «карта». Типи даних для часу, інтервалів часу, символьних рядків з двобайтовим поданням символів були додані в мову SQL. З'явилися технології DataMining, сховища даних, мультимедійні бази даних і web-бази даних.

Виникнення і розвиток Data Mining зумовлене різними факторами, основними серед яких є такі:

- вдосконалення апаратного і програмного забезпечення;
- вдосконалення технологій зберігання і запису даних;
- накопичення великої кількості ретроспективних даних;
- вдосконалення алгоритмів обробки інформації.

У 1993 році вийшла перша розсилка «Knowledge Discovery Nuggets», а в 1994 році був створений один з перших сайтів по data mining [2].

Data Mining широко використовується в багатьох областях з великим об'ємом даних. В науці - астрономії, біології, біоінформатиці, медицини, фізики та інших областях. У бізнесі - торгівлі, телекомунікація, банківській справі, промисловому виробництві і т. Д. Завдяки мережі Інтернет Data Mining використовується кожен день тисячі разів в секунду - кожен раз, коли хтось використовує Гугл (Google) або інші пошукові системи (search engines) на просторах Інтернету.

Види інформації, з якими працюють дослідники, включають не тільки цифрові дані, але і все більш текст, зображення, відео, звук і т. Д. Одна нова і швидко зростаюча частина Data Mining – це аналіз зв'язків між даними (link analysis) , яка має додатки в таких різних областях, як біоінформатика, цифрові бібліотеки і захист проти тероризму.

Математичний і статистичний підходи є основою для Data Mining.

Термін Data Mining часто перекладається як видобуток даних, вилучення інформації, розкопка даних, інтелектуальний аналіз даних, засоби пошуку закономірностей, витяг знань, аналіз шаблонів, «витяг зерен знань з гір даних», розкопка знань в базах даних, інформаційна проходка даних, «промивання» даних [8]. Поняття «виявлення знань в базах даних» (knowledge discovery in databases, KDD) можна вважати синонімом Data Mining. Поняття Data Mining, що з'явилося в 1978 р, набуло високу популярність в сучасному трактуванні приблизно з першої половини 90-х років. До цього часу обробка і аналіз даних здійснювалися в рамках прикладної статистики, при цьому в основному вирішувалися завдання обробки невеликих баз даних. Про популярність Data Mining говорить і той факт, що результат пошуку терміна «Data Mining» в пошуковій система Яндекс (на січень 2018 роки) - більше 58 мільйонів результатів.

«Data Mining» - є мультидисциплінарної областю, яка включає в себе такі поняття як Статистика, Штучний інтелект, Машинне навчання, алгоритмізації, Розпізнавання образів, Теорію БД, Візуалізацію та інші дисципліни.

Статистика – це наука про методи збору даних, їх обробки і аналізу для виявлення закономірностей, властивих досліджуваному явищу. Статистика є сукупністю методів планування експерименту, збору даних, їх подання та узагальнення, а також аналізу і отримання висновків на підставі цих даних. Статистика оперує даними, отриманих в результаті спостережень або експериментів. Одна з наступних глав буде присвячена поняттю даних.

Штучний інтелект – науковий напрям, в рамках якого ставляться і вирішуються завдання апаратного або програмного моделювання видів людської діяльності, що традиційно вважаються інтелектуальними. Термін інтелект (intelligence) походить від латинського intellectus, що означає розум, розум, розум, розумові здібності людини. Відповідно, штучний інтелект (AI, artificial intelligence) тлумачиться як властивість автоматичних систем брати на себе окремі функції інтелекту людини. Штучним інтелектом називають властивість інтелектуальних систем виконувати творчі функції, які традиційно вважаються прерогативою людини. Кожен з напрямків, які сформували Data Mining, має свої особливості. Проведемо порівняння з деякими з них.

Єдиного визначення машинного навчання на сьогоднішній день немає. Машинне навчання можна охарактеризувати як процес отримання програмою нових знань. Мітчелл в 1996 році дав таке визначення: «Машинне навчання – це наука, яка вивчає комп'ютерні алгоритми, автоматично покращуючись під час роботи». Одним з найбільш популярних прикладів алгоритму машинного навчання є нейронні мережі.

Поняття Data Mining тісно пов'язане з технологій баз даних та поняттям дані.

Актуальність інтелектуального аналізу даних зростає все більше, фахівці в цій галузі дуже затребувані на ринку праці. Дана тема цікава у виявленні прихованих правил і закономірностей в наборах даних. Справа в тому, що людський розум сам по собі не пристосований для сприйняття великих масивів різнорідної інформації. Людина до того ж не здатний вловлювати більше двох-трьох взаємозв'язків навіть у невеликих вибірках. Але і традиційна математична статистика, довгий час претендувала на роль основного інструменту аналізу даних, також нерідко пасує при вирішенні завдань з реального складного життя. Вона оперує усередненими характеристиками вибірки, які часто є фіктивними величинами (типу середньої температури пацієнтів у лікарні, середньої висоти будинку на

вулиці, що складається з палаців і халуп і т.п.). Тому методи математичної статистики виявляються корисними головним чином для перевірки заздалегідь сформульованих гіпотез.

В принципі немає нічого нового в постановці завдання Data Mining. Фахівці протягом декількох останніх десятків років вирішували подібні завдання («пошук емпіричних закономірностей», «евристичний пошук в складних середовищах», «індуктивний висновок» і т. П.). Але тільки зараз суспільство в цілому дозріло для розуміння практичної важливості і широти цих завдань. По-перше, у зв'язку з розвитком технологій запису і зберігання даних сьогодні на людей обрушилися колосальні потоки інформаційної руди в самих різних областях, які без продуктивної переробки загрожують перетворитися в нікому не потрібні звалища. І, по-друге, засоби і методи обробки даних стали доступними і зручними, а їх результати зрозумілими будь-якій людині.

## 1.2 Основні проблеми та їх вирішення

Основним завданням аналітика є генерація гіпотез. Він вирішує її, ґрунтуючись на своїх знаннях і досвіді. Однак знання є не тільки у людини, але і в накопичених даних, які піддаються аналізу. Такі знання часто називають «прихованими», так як вони містяться в гігабайтах і терабайт інформації, які людина не в змозі дослідити самостійно. У зв'язку з цим існує висока ймовірність пропустити гіпотези, які можуть принести значну вигоду.

Очевидно, що для виявлення прихованих знань необхідно застосовувати спеціальні методи автоматичного аналізу, за допомогою яких доводиться практично здобувати знання з «завалів» інформації. За цим напрямком міцно закріпився термін видобуток даних або Data Mining. Класичне визначення цього терміну дав в 1996 р один із засновників цього направлення Пятецкий-Шапіро.

Сфера застосування Data Mining нічим не обмежена – вона скрізь, де є які-небудь дані. Але в першу чергу методи Data Mining сьогодні, м'яко кажучи, заінтригували комерційні підприємства, що розгортають проекти на основі інформаційних сховищ даних (Data Warehousing). Досвід багатьох таких підприємств показує, що віддача від використання Data Mining може досягати 1000%. Наприклад, відомі повідомлення про економічний ефект, в 10-70 разів перевищив початкові витрати від 350 до 750 тис. дол. [3]. Наводяться відомості про проект в 20 млн. Дол., Який окупився всього за 4 місяці. Інший приклад – річна економія 700 тис. дол. за рахунок впровадження Data Mining в мережі універсамів у Великобританії.

Data Mining представляють велику цінність для керівників і аналітиків в їх повсякденній діяльності. Ділові люди усвідомили, що за допомогою методів Data Mining вони можуть отримати відчутні переваги в конкурентній боротьбі. Коротко охарактеризуємо деякі можливі бізнес-додатки Data Mining [3].

Будувати бізнес без аналізу – все одно, що ходити по обриву наосліп. Аналіз допомагає зрозуміти, які товари продаються краще, а які гірше, і чому. Також з його допомогою ви можете побачити, які опції ресурсу вимагають доопрацювання, а з якими і зовсім можна розпрощатися.

Для чого необхідний аналіз сайту Інтернет-магазину? Перш за все, аналізуючи отриману інформацію, Ви зможете оптимально налаштувати свій магазин для продажу – це буде зручно вашим клієнтам, і прибутково для Вас.

Раніше найпростішим кроком було використання установки лічильника відвідуваності на сторінки. При цьому, Ви одним пострілом отримували 2 зайця: отримували статистику відвідуваності по лічильнику і рекламувалися в потрібній категорії рейтингу, куди Ви внесли свій Інтернет-магазин. Єдиним мінусом цього варіанту – необхідність картинки лічильника на всіх сторінках.

Більшість початківців ритейлерів відчувають розчарування через мізерно малих обсягів продажів. Брак інформації та знань вузького профілю призводить до того, що продавці роблять помилки ще при «зародження» сайту. Але не варто засмучуватися. Основні проблеми інтернет-магазинів однотипні, в будь-якому випадку їх нескладно виявити і усунути, використовуючи технології «Data Mining».

Цікавий варіант отримання статистики – аналіз лог-файлів на вашому сервері.

Лог-файл – ця інформація по роботі вашого сайту, яка пишеться сервером. Використовуючи спеціальні програми на сервер ви отримуєте повну картину активності користувачів. Для цього використовуються Лог-аналізатор.

## 1.3 Актуальність теми

Згідно зі статистикою застосування, Data Mining найчастіше використовується в наступних областях:

•електронна комерція;

• телекомунікації;

•керування виробництвом;

• біоінформатика (генні дослідження);

•банківські послуги;

• страхування;

• взаємини з клієнтами (CRM);

• виявлення випадків шахрайства;

• маркетинг;

• інвестування.

У системах електронного бізнесу, де особливу важливість мають питання залучення і утримання клієнтів, технології Data Mining часто застосовуються для побудови рекомендаційних систем інтернет-магазинів і для вирішення проблеми персоналізації відвідувачів Web-сайтів. Персоналізація клієнтів, тобто автоматичне розпізнавання приналежності клієнта до певної цільової аудиторії дозволяє компанії проводити більш гнучку маркетингову політику.

Для успішного просування товарів завжди важливо знати, що і як продається, а також хто є споживачем. Знаючи зв'язку між покупками і часові закономірності, можна оптимальним чином регулювати пропозицію. Data Mining дозволяє вирішувати завдання виділення груп споживачів зі схожими стереотипами поведінки, тобто сегментувати ринок. Для цього можна застосовувати такі технології Data Mining, як кластеризація і класифікація.

Телекомунікаційний бізнес є однією з найбільш динамічно розвиваються областей сучасної економіки.

Телекомунікаційні компанії працюють в умовах жорсткої конкуренції. При цьому відомо, що утримати клієнта набагато дешевше, ніж залучити

нового, а ось повернути старого клієнта буде коштувати у багато разів більше, ніж його утримати. Тут, як і всюди в економіці, справедливо правило Парето – тільки 20% клієнтів приносять компанії 80% доходу [4]. Крім цього, існує ряд клієнтів, що завдають компанії прямої шкоди. Через випадки шахрайства в телекомунікаційній індустрії в рік втрачається близько 4 млрд доларів. Таким чином, використання технологій Data Mining, спрямованих як на аналіз прибутковості, так і на захист від шахрайства, заощадить компанії величезні кошти. Ще один з поширених способів використання методів Data Mining в області телекомунікацій – це аналіз записів про докладних характеристиках викликів. Призначення такого аналізу - виявлення категорій клієнтів зі схожими стереотипами користування послугами та розробка привабливих наборів цін і послуг.

Прикладом застосування Data Mining в промисловості може бути прогнозування якості виробу в залежності від заміряються параметрів технологічного процесу. Подібні ж методи використовуються для прогнозування відмов обладнання на підставі незначних змін його характеристик (вібрацій, теплового режиму та ін.) У медичних і біологічних дослідженнях, а також у практичній медицині, спектр вирішуваних завдань настільки широкий, що можливе використання будь-яких методологій Data Mining. Прикладом може служити побудова діагностичної системи або дослідження ефективності лікування захворювання. Відомо багато експертних систем для постановки медичних діагнозів. Вони побудовані на основі правил, що описують поєднання різних симптомів окремих захворювань. За допомогою таких правил можна дізнатися не тільки саме захворювання, а й те, як потрібно його потрібно лікувати. Правила допомагають вибирати засоби медикаментозного впливу, визначати показання та протипоказання, орієнтуватися в лікувальних процедурах, створювати умови найбільш ефективного лікування, прогнозувати результати призначеного курсу лікування і т. П. Технології Data Mining дозволяють виявляти в медичних даних шаблони, що становлять основу

зазначених правил. Одним з найбільш передових напрямків медицини є біоінформатика – область науки, що розробляє і застосовує обчислювальні алгоритми для аналізу і систематизації генетичної інформації з метою з'ясування структури і функції макромолекул, подальшого використання цих знань для пояснення різних біологічних явищ і створення нових лікарських препаратів.

Data Mining використовує ряд ідей і технічних досягнень технології OLAP і може розглядатися як одна з її альтернатив. Якщо в OLAP користувачеві необхідно самому формулювати гіпотези про неявно присутніх фактах, то в Data Mining застосовується автоматизований процес дослідження великих масивів даних, вироблений з метою виявлення прихованих закономірностей (знань). Закономірності, з точки зору використання Data Mining, повинні бути нетривіальними і практично корисними для прийняття рішень в різних сферах людської діяльності. Дані зазвичай беруться зі сховищ даних [4], а не зі звичайних оперативних баз. В основі концепції сховищ даних лежить ідея поділу даних, що використовуються для оперативного опрацювання та для вирішення задач аналізу. У. Інмона в своїй монографії «Побудова сховищ даних» докладно описав цю концепцію і дав наступне визначення: «сховище даних - предметно-орієнтований, інтегрований, незмінність, що підтримує хронологію набір даних, організований для цілей підтримки прийняття рішень».

Актуальність даного напрямку зростає в зв'язку з тим, що в даний час багато компаній надають доступ до своїх інформаційних ресурсів. Це відноситься як до інформації в вигляді HTML-сторінок, так і до даних з баз даних компаній, корпоративних порталів та ін. Безумовно, залишається частина даних, до яких доступ неможливий з міркувань конфіденційності. Особливістю Web-ресурсів є різнорідність представленої інформації: текстові файли, зображення, звук, відео, метадані, а також гіперпосилання. У зв'язку з цим технологія Web Mining тісно пов'язана з іншими напрямами

Data Mining. Так, для аналізу текстової інформації використовуються методи Text Mining (див. Розділ 5). Для аналізу зображень, відео- та аудіо-інформації використовується Multimedia Mining. Витяг Web-контенту дозволяє підвищити релевантність одержуваної інформації і використовується для індексації документів. Крім того, воно забезпечує структуризацію інформації, розширюючи можливості запитів до неї. Під час вилучення Web-структур будуються моделі, що відображають взаємозв'язки між Web-сторінками. Модель ґрунтується на топології гіперпосилань. Така модель може використовувати категоризацію Web-сторінок і бути корисна для генерації інформації про ставлення і подібності між Web-сайтами. Дана категорія Web Mining може бути використана для розпізнавання авторських сайтів і оглядових сайтів за темами.

# РОЗДІЛ 2. АНАЛІЗ НАПРЯМКУ ДОСЛІДЖЕННЯ ТА ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 2.1 Інтелектуальний аналіз даних

Data mining (рус. Видобуток даних, інтелектуальний аналіз даних, глибинний аналіз даних) – збірна назва, що використовується для позначення сукупності методів виявлення в даних раніше невідомих, нетривіальних, практично корисних і доступних інтерпретації знань, необхідних для прийняття рішень в різних сферах людської діяльності [ 2]. Термін введений Григорієм Пятецкий-Шапіро в 1989 році.

Витрачені зусилля на відкриття знань, які вже відомі користувачеві, не окупаються. Цінність представляють саме нові, раніше невідомі знання.

Суть і мета технології Data Mining можна охарактеризувати так: це технологія, яка призначена для пошуку у великих обсягах даних неочевидних, об'єктивних і корисних на практиці закономірностей.

Неочевидних – це значить, що знайдені закономірності не виявляються стандартними методами обробки інформації або експертним шляхом.

Об'єктивних – це значить, що виявлені закономірності будуть повністю відповідати дійсності, на відміну від експертної думки, яке завжди є суб'єктивним.

Практично корисних – це значить, що висновки мають конкретне значення, якому можна знайти практичне застосування.

Знання – сукупність відомостей, яка утворює цілісний опис, відповідне деякому рівні обізнаності про описуваному питанні, предметі, проблеми і т.д. Використання знань (knowledge deployment) означає дійсне застосування знайдених знань для досягнення конкретних переваг (наприклад, в конкурентній боротьбі за ринок).

В основу технології Data Mining покладена концепція шаблонів (patterns), які представляють собою закономірності, властиві підвибірках даних, котрі можуть бути виражені у формі, зрозумілою людині.

«Mining» по-англійськи означає "видобуток корисних копалин", а пошук закономірностей у величезній кількості даних дійсно схоже на цього процесу.

Завдання (tasks) Data Mining іноді називають закономірностями (regularity) або техніками (techniques).

Єдиної думки щодо того, які завдання слід відносити до Data Mining, немає. Більшість авторитетних джерел перераховують наступні: класифікація, кластеризація, прогнозування, асоціація, візуалізація, аналіз і виявлення відхилень, оцінювання, аналіз зв'язків, підведення підсумків.

Завдання класифікації зводиться до визначення класу об'єкта по його характеристики. Необхідно зауважити, що в цьому завданні безліч класів, до яких може бути віднесений об'єкт, заздалегідь відомо.

Завдання регресії, подібно задачі класифікації, дозволяє визначити за відомими характеристиками об'єкта значення деякого його параметра. На відміну від завдання класифікації значенням параметра є не кінцеве безліч класів, а множина дійсних чисел.

При пошуку асоціативних правил метою є знаходження частих залежностей (або асоціацій) між об'єктами або подіями. Най-денние залежності представляються у вигляді правил і можуть бути використано як для кращого розуміння природи аналізованих даних, так і для передбачення появи подій. Завдання кластеризації полягає в пошуку незалежних груп (класти-рів) і їх характеристик у всьому безлічі аналізованих даних. Рі-ня цього завдання допомагає краще зрозуміти дані. Крім того, угрупування однорідних об'єктів дозволяє скоротити їх число, а отже, і полегшити аналіз.

Перераховані завдання за призначенням поділяються на описові і передбачальні.

Описові (descriptive) завдання приділяють увагу поліпшенню розуміння аналізованих даних. Ключовий момент у таких моделях – легкість і прозорість результатів для сприйняття людиною. Можливо, виявлені закономірності будуть специфічною рисою саме конкретних досліджуваних

даних і більше ніде не зустрінуться, але це все одно може бути корисно і тому має бути відомо. До такого виду завдань відносяться кластеризація і пошук асоціативних правил.

Рішення передбачальніх (predictive) завдань розбивається на два етапи. На першому етапі на підставі набору даних з відомими результатами будується модель. На другому етапі вона використовується для передбачення результатів на підставі нових наборів даних. При цьому, природно, потрібно, щоб побудовані моделі працювали максимально точно. До даного виду завдань відносять завдання класифікації і регресії. Сюди можна віднести і завдання пошуку асоціативних правил, якщо результати її рішення можуть бути використані для передбачення появи деяких подій.

Для успішного просування товарів завжди важливо знати, що і як продається, а також, хто є споживачем. Вичерпну відповідь на перше запитання дають такі кошти Data Mining, як аналіз ринкових кошиків і сіквенціальний аналіз. Знаючи зв'язку між покупками і часові закономірності, можна оптимальним чином регулювати пропозицію. З іншого боку, маркетинг має можливість безпосередньо управляти попитом, але для цього необхідно знати якомога більше про споживачів – цільової аудиторії маркетингу. Data Mining дозволяє вирішувати завдання виділення груп споживачів зі схожими стереотипами поведінки, т. Е. Сегментувати ринок. Для цього можна застосовувати такі технології Data Mining, як кластеризацию і класифікацію.

У даній роботі розглядається аналіз даних інтернет магазину і, вивчивши моделі аналізу даних, зроблено логічне висновок, що для застосування методів Data Mining необхідно застосувати класичну задачу кластеризації даних, яка є описової завданням і навчається без учителя.

Описові моделі приділяють увагу суті залежностей в наборі даних, взаємному впливу різних чинників, т. Е. На побудові емпірічних моделей різних систем. Ключовий момент у таких моделях – легкість і прозорість для сприйняття людиною. Можливо, виявлені закономірності будуть

специфічною рисою саме конкретних досліджуваних даних і більше ніде не зустрінуться, але це все одно може бути корисно і тому має бути відомо.

Кластеризації – описують групи (кластери), на які можна поділити об'єкти, дані про яких піддаються аналізу. Групуються об'єкти (спостереження, події) на основі даних (властивостей), які описують сутність об'єктів. Об'єкти усередині кластера повинні бути «схожими» один на одного і відрізнятися від об'єктів, які увійшли в інші кластери. Чим більше схожі об'єкти усередині кластера і чим більше відмінностей між кластерами, тим точніше кластеризація.

Технологія Data Mining розвивалася і розвивається на стику таких дисциплін як статистика, теорія інформації, машинне навчання, теорія баз даних, цілком закономірно, що більшість алгоритмів і методів Data Mining були розроблені на основі різноманітних технологій і концепцій.

Завдання кластеризації полягає в поділі досліджуваної множини об'єктів на групи «схожих» об'єктів, які називаються кластерами. Слово кластер англійського походження (cluster), перекладається як згусток, пучок, група. Споріднені поняття, використовувані в літературі, - клас, таксон, згущення. Часто рішення задачі розбиття множини елементів на кластери називають кластерним аналізом.

Кластеризація може застосовуватися практично в будь-якій області, де необхідно дослідження експериментальних або статистичних даних.

Для наукових досліджень вивчення результатів кластеризації, а саме з'ясування причин, за якими об'єкти об'єднуються в групи, здатне відкрити нові перспективні напрямки. Традиційним прикладом, який зазвичай призводять для цього випадку, є періодична таблиця елементів. У 1869 р Дмитро Менделєєв розділив 60 відомих в той час елементів на кластери або періоди. Елементи, що потрапили в одну групу, володіли схожими характеристиками. Вивчення причин, за якими елементи розбивалися на явно виражені кластери, в значній мірі визначив пріоритети наукових досліджень

на роки вперед. Але лише через 50 років квантова фізика дала переконливі пояснення періодичної системи.

Кластеризація відрізняється від класифікації тим, що для проведення аналізу не потрібно мати виділену залежну змінну. З цієї точки зору вона відноситься до класу unsupervised learning. Це завдання вирішується на початкових етапах дослідження, коли про дані мало що відомо. Її рішення допомагає краще зрозуміти дані, і з цієї точки зору завдання кластеризації є описової завданням.

Для завдання кластеризації характерна відсутність будь-яких відмінностей як між змінними, так і між об'єктами. Навпаки, шукаються групи найбільш близьких, схожих об'єктів. Методи автоматичного розбиття на кластери рідко використовуються самі по собі, просто для отримання груп схожих об'єктів. Після визначення кластерів застосовуються інші методи Data Mining, для того щоб спробувати встановити, а що означає таке розбиття, чим воно викликане.

Кластерний аналіз дозволяє розглядати досить великий обсяг інформації і різко скорочувати, стискати великі масиви інформації, робити їх компактними і наочними.

Відзначимо ряд особливостей, властивих завданню кластеризації.

По-перше, рішення сильно залежить від природи об'єктів даних (і їх атрибутів). Так, з одного боку, це можуть бути однозначно визначені, чітко кількісно окреслені об'єкти, а з іншого – об'єкти, які мають розподіл усіх або нечітке опис.

По-друге, рішення значно залежить також і від уявлення кластерів і передбачуваних відносин об'єктів даних і кластерів. Так, необхідно враховувати такі властивості, як можливість / неможливість приналежності об'єктів декільком кластерам. Необхідно визначення самого поняття приналежності кластеру: однозначна (належить / не належить), імовірнісна (ймовірність приналежності), нечітка (ступінь приналежності).

При виконанні кластеризації важливо, скільки в результаті повинно бути побудовано кластерів. Передбачається, що кластеризація повинна виявити природні локальні згущення об'єктів. Тому число кластерів є параметром, часто істотно ускладнює вид алгоритму, якщо передбачається невідомим, та суттєво впливають на якість результату, якщо вони відомі.

Проблема вибору числа кластерів вельми нетривіальна. Досить сказати, що для отримання задовільного теоретичного рішення часто необхідно зробити досить сильні припущення про властивості деякого заздалегіть заданого сімейства розподілів. Тому алгоритми кластеризації зазвичай будуються як деякий спосіб перебору числа кластерів і визначення його оптимального значення в процесі перебору.

Число методів розбиття множини на кластери досить велике. Всі їх можна поділити на ієрархічні і неієрархічні.

У неієрархічних алгоритмах характер їх роботи і умова зупинки необхідно заздалегідь регламентувати часто досить великим числом параметрів, що іноді важко, особливо на початковому етапі вивчення матеріалу. Але в таких алгоритмах досягається велика гнучкість в варіюванні кластеризації і зазвичай визначається число кластерів.

З іншого боку, коли об'єкти характеризуються великим числом ознак (параметрів), то набуває важливого значення завдання угруповання ознак. Вихідна інформація міститься в квадратній матриці зв'язків ознак, зокрема в кореляційної матриці. Основою успішного вирішення завдання угруповання є неформальна гіпотеза про невелике число прихованих чинників, які визначають структуру взаємних зв'язків між ознаками.

В ієрархічних алгоритмах фактично відмовляються від визначення числа кластерів, будуючи повне дерево вкладених кластерів (Дендрограмма). Число кластерів визначається з припущень, в принципі, не відносяться до роботи алгоритмів, наприклад по динаміці зміни порога розщеплення (злиття) кластерів. Труднощі таких алгоритмів добре вивчені: вибір заходів близькості кластерів, проблема інверсій індексації в Дендрограма,

негнучкість ієрархічних класифікацій, яка іноді вельми небажана. Проте, уявлення кластеризації у вигляді дендрограми дозволяє отримати найбільш повне уявлення про структуру кластерів.

Ієрархічні алгоритми пов'язані з побудовою дендрограм і діляться:

а) на агломеративні, що характеризуються послідовним об'єднанням вихідних елементів і відповідним зменшенням числа кластерів (побудова кластерів від низу до верху);

б) на дівізімние (подільні), в яких число кластерів зростає починаючи з одного, в результаті чого утворюється послідовність розщеплюють груп (побудова кластерів зверху вниз).

Генетичні алгоритми (ГА) відносяться до числа універсальних методів оптимізації, що дозволяють вирішувати завдання різних типів (комбінаторні, спільні завдання з обмеженнями і без обмежень) і різного ступеня складності. При цьому ГА характеризуються можливістю як однокрітеріального, так і багатокритеріального пошуку в значній території, ландшафт якого є негладким.

В останні роки різко зросла кількість робіт, перш за все зарубіжних вчених, присвячених розвитку теорії ГА і питань їх практичного використання. Результати даних досліджень показують, зокрема, що ГА можуть отримати більш широке поширення при інтеграції з іншими методами і технологіями. З'явилися роботи, в яких доводиться ефективність інтеграції ГА і методів теорії нечіткості, а також нейронних обчислень і систем.

Ефективність такої інтеграції знайшла практичне підтвердження в раз-розробці відповідних інструментальних засобів (ІС). Так, фірма Attar Software включила ГА-компонент, орієнтований на вирішення завдань опти-мізації, в свої ІС, призначені для розробки експертної системи. Фірма California Scientific Software зв'язала ІС для нейронних мереж з ГА-компонентами, що забезпечують автоматичну генерацію і настрой¬ку нейронної мережі. Фірма NIBS Inc. включила в свої ІС для нейронних сітей,

орієнтовані на прогнозування ринку цінних паперів, ГА-компоненти, які, на думку фінансових експертів, дозволяють уточнювати прогнозування.

Незважаючи на відомі загальні підходи до такої інтеграції ГА і нечіткої логіки, як і раніше актуальна задача визначення найбільш значимих параметрів операційного базису ГА з метою їх адаптації в процесі роботи ГА за рахунок використання нечіткого продукційного алгоритму (НПА).

Перераховані нижче причини комерційного успіху інструментальних засобів в області штучного інтелекту можуть розглядатися як загальні вимоги до розробки систем аналізу даних, що використовуються ГА:

інтегрованість - розробка ІС, легко інтегруються з іншими інформаційними технологіями і засобами;

відкритість і переносимість - розробка ІС відповідно до стандартами, що забезпечують можливість виконання в неоднорідному програмно-апаратному оточенні і переносимість на інші платформи без перепрограмування;

використання мов традиційного програмування. Перехід до ІС, реалізованим на мовах традиційного програмування (С, С ++ і т. Д.), Спрощує забезпечення інтегрованості, знижує вимоги додатків до швидкодії ЕОМ і обсягами оперативної пам'яті;

архітектура «клієнт-сервер» - розробка ІС, що підтримують розбрати-поділені обчислення в архітектурі «клієнт-сервер», що дозволяє знизити вартість обладнання, що використовується в додатках, децентралізувати додатки і підвищити їх продуктивність.

Перераховані вимоги обумовлені необхідністю створення інтегрованого-рова додатків, т. Е. Додатків, які об'єднують в рамках єдиного комплексу традиційні програмні системи з системами штучного інтелекту і ГА зокрема.

Інтеграція ГА і нейронних мереж дозволяє вирішувати проблеми пошуку оптимальних значень ваг входів нейронів, а інтеграція ГА і нечіткої

логіки дозволяє оптимізувати систему продукційних правил, які можуть бути використані для управління операторами ГА (двунаправлена інтеграція).

Одним з найбільш затребуваних додатків ГА в області Data Mining є пошук найбільш оптимальної моделі (пошук алгоритму, специфіці конкретної області).

Нейронні мережі - це клас моделей, заснованих на біологічної аналогії з мозком людини і призначених після проходження етапу так називаного навчання на наявних даних для вирішення різноманітних завдань аналізу даних. При застосуванні цих методів перш за все постає питання вибору конкретної архітектури мережі (числа «шарів» і кількості «нейронів» в кожному з них). Розмір і структура мережі повинні відповідати (наприклад, в сенсі формальної обчислювальної складності) суті досліджуваного явища. Оскільки на початковому етапі аналізу природа явлення зазвичай відома погано, вибір архітектури є непростим завданням і часто пов'язаний з тривалим процесом «проб і помилок» (проте останнім часом стали з'являтися нейронно-мережеві програми, в яких для рішення трудомісткою завдання пошуку найкращою архітектури мережі застосовуються методи штучного інтелекту).

Потім побудована мережа піддається процесу так званого навчання. На цьому етапі нейрони мережі ітеративно обробляють вхідні дані і коректують свої ваги так, щоб мережа найкращим чином прогнозувала (в традиційних термінах варто було б сказати «здійснювала підгонку») дані, на яких виконується «навчання». Після навчання на наявних даних мережа готова до роботи і може використовуватися для побудови прогнозів.

Нейронна мережа, отримана в результаті «навчання», висловлює закономірності, присутні в даних. При такому підході вона виявляється функціональним еквівалентом певної моделі залежностей між змінними, подібної тим, які будуються в традиційному моделюванні. Однак, на відміну від традиційних моделей, в разі нейронних мереж ці залежності не можуть бути записані в явному вигляді, подібно до того як це робиться в статистиці.

Іноді нейронні мережі видають прогноз дуже високої якості; проте вони являють собою типовий приклад нетеоретична підходу до дослідження (іноді це називають «чорним ящиком»). При такому підході зосереджуються виключно на практичний результат, в даному випадку на точності прогнозів і їх прикладної цінності, а не на суті механізмів, що лежать в основі явища або відповідно отриманих результатів будь-якої наявної теорії.

Слід, однак, відзначити, що методи нейронних мереж можуть застосовуватися і в дослідженнях, спрямованих на побудову пояснює моделі явища, оскільки нейронні мережі допомагають вивчати дані з метою пошуку значущих змінних або груп таких змінних, і отримані результати мо¬гут полегшити процес подальшого побудови моделі. Більш того, зараз є нейромережеві програми, які за допомогою складних алгоритмів можуть знаходити найбільш важливі вхідні змінні, що вже безпосередньо допомагає будувати модель.

Одне з головних переваг нейронних мереж полягає в тому, що вони, принаймні теоретично, можуть апроксимувати будь-яку безперервну функцію, і тому досліднику немає необхідності заздалегідь приймати будь-які гіпотези щодо моделі і навіть, в ряді випадків, про те, які змінні дійсно важливі. Однак суттєвим недоліком нейронних мереж є та обставина, що остаточне рішення залежить від початкових установок мережі та, як уже зазначалося, його практично неможливо інтерпретувати в традиційних аналітичних термінах, котрі зазвичай застосовуються при побудові теорії явища.

Деякі автори відзначають той факт, що нейронні мережі використовують, або, точніше, припускають використання обчислювальних систем з масовим паралелізмом.

Завдання кластеризації полягає в поділі досліджуваної множини об'єктів на групи схожих об'єктів, званих кластерами.

Для визначення «схожост» об'єктів вводиться міра близькості, названа відстанню. Існують різні способи обчислення відстаней: Евклідово, Манхеттенський, Чебишева та ін.

Результати кластеризації можуть бути представлені різними способами. Одним з найбільш популярних є дентограмма - відображення по-отже процесу кластеризації.

Базові методи кластеризації діляться на ієрархічні і неієрархічні. Перші будують дентограмми або від низу до верху (агломеративні), або зверху вниз (дивізимні).

Найбільш популярний з неієрархічних алгоритмів - алгоритм к-середній і його різновиди. Ідея методу полягає у визначенні центрів до кластерів та віднесення до кожного кластеру об'єктів, найближче знаходяться до цих центрів.

Побудовано відношення а-толерантності на безлічі зразків даних. Запропонована конструктивна процедура побудови даного відносини відображає інтуїтивне розуміння того, як можна порівняти два зразка даних в рамках запропонованого безлічі зразків. Зазначене відношення виходить з наступного правила: «якщо два зразка даних подібні щодо кожного із зразків даних вхідного безлічі, то їх можна вважати подібними щодо всієї множини зразків даних».

Пропонується процедура отримання цього відносини з відносних заходів подібності зразків даних, які в свою чергу виходять на основі нормальної міри схожості і поняття відстані між зразком даних.

Побудовано відношення а-квазіеквівалентності на безлічі зразків даних. Отримання даного відносини засноване на побудові транзитивних зв'язків на відносно а-толерантності. Математична процедура, покладена в основу обчислення даного відносини (насамперед методика обчислення транзитивного замикання на відносно а-толерантності), має на увазі під собою наступну семантику: "ступінь подібності двох зразків даних щодо а-квазіеквівалентності може збільшитися по відношенню до ступеня подібності

цих зразків в відношення а-толерантності, якщо між ними знайдеться така послідовність зразків даних, ступінь подібності між сусідніми зразками даних в якій перевищує вихідну ступінь схожості між заданими зразками на всій послідовності зразків даних». Таким чином, ставлення а-квазіеквівалентності є безпосереднім інструментом кластеризації даних. Дане відношення може бути легко перетворено в відношення еквівалентності в класичному сенсі, а це дозволяє розбивати безліч зразків даних на класи еквівалентності, що є, по суті, основним результатом розв'язання задачі кластерізаціі.

Відповідно до постановкою завдання кластеризації її рішення в рамках запропонованого підходу виглядає наступним чином:

• спосіб порівняння даних - в даному виданні запропоновано кілька заходів подібності, що володіють різним ступенем об'єктивності по відношення до порівнюваним об'єктам. У наведеному далі списку кожний наступний із способів порівняння даних ґрунтується на попередньому і є більш відповідним для безпосереднього рішення завдання кластеризації;

• порівняння по відстані між зразками даних (в прикладах, наведених в даному виданні, використовується Евклідова відстань) - даний спосіб порівняння можна охарактеризувати як попарне порівняння зразків даних в термінах тих одиниць виміру, в яких задаються атрибути зразків даних;

• порівняння за допомогою сімейства нормальних заходів подібності - результати порівняння за допомогою даного способу є безрозмірними числовими значеннями в діапазоні від 0 до одиниці і можуть інтерпретувати як нечіткі множини зразків даних, близьких до кожного із зразків даних. Цей спосіб порівняння можна охарактеризувати як почергове порівняння всіх зразків даних з кожним із зразків;

• порівняння за допомогою сімейства відносних заходів подібність--

результатом порівняння за допомогою даного способу є сімейство нечітких відносин а-толерантності, кожне з яких має сенс попарного порівняння зразків даних щодо заданого зразка. Результати порівняння

виражаються безрозмірною величиною в діапазоні від 0 до 1. Чим більше значення цієї величини, тим більш

схожі елементи.

Математична інтерпретація відносної міри схожості - ступінь приналежності кожної пари зразків даних до відповідного відношенню подібності;

• порівняння зразків даних за допомогою відносини а-толерантності на безлічі зразків даних-перший в даному списку спосіб «об'єктивного» порівняння зразків даних, т. К. Враховує схожість будь-яких двох зразків даних щодо всіх інших зразків. Цей спосіб порівняння даних ще не можна безпосередньо використовувати для їх кластеризації, т. К. В відношенні а-толерантності відсутній властивість транзитивності, т. Е. Відсутня можливість групового порівняння даних. Результатом порівняння двох зразків даних є безрозмірна величина в діапазоні від нуля до одиниці. Математична інтерпретація цього заходу схожість - ступінь приналежності кожної пари зразків даних до відношенню а-толірантності;

• порівняння за допомогою відносини а-квазіеквівалентності і шкали відносини а-квазіеквівалентності - це той спосіб порівняння зразків даних, який безпосередньо використовується для кластеризації даних. На відміну від попереднього цей спосіб порівняння є найбільш "об'єктивним", т. К. В відношенні а-квазіеквівалентності присутній один з нечітких аналогів властивості транзитивності (щодо а-квазітранзітівності), що дозволяє враховувати міжгрупове схожість даних. Використовуючи мінімально повний набір рівнів відносини а-квазіеквівалентності, можна породити ціле сімейство відносин еквівалентності в класичному сенсі, за допомогою яких будується набір розбиття безлічі зразків даних на класи еквівалентності. Результатом порівняння є безрозмірна величина в діапазоні від 0 до 1. Математична інтерпретація даної міри схожості - ступінь приналежності кожної пари зразків даних до відношенню а-квазіеквівалентності;

• спосіб кластеризації - способом кластеризації в даному виданні відповідно до обраного підходом є застосування сімейства відносин еквівалентності. Кожне з цих відносин виходить за допомогою переходу від ставлення а-квазіеквівалентності до відношення еквівалентності в класичному сенсі використанням відповідного рівня відносини а-квазіеквівалентності з шкали відносини а-квазіеквівалентності. Ті зразки даних, які в відповідності зі ставленням а-квазіеквівалентності мають схожість, що перевищує зазначений рівень, є еквівалентними, осталь¬ние - нееквівалентними;

• розбиття даних по кластерам - розбиття по кластерам НЕ є однозначним. Це очевидно з припущення про нечіткої взаємозв'язку даних. Проте кількість розбиття відповідно до даного підходу є кінцевим і визначається потужністю шкали відношення а-квазіеквівалентності. Кожне конкретне розбиття по кластерам відповідає розбиття безлічі зразків даних на класи еквівалентності при даному рівні а-квазіеквівалентності.

Більш коротко рішення задачі кластеризації виглядає так:

• спосіб порівняння - відношення а-квазіеквівалентності і його шкала;

• спосіб кластеризації - застосування сімейства відносин еквівалентності, одержуваного з відносини а-квазіеквівалентності і шкали відносини а-квазіеквівалентності;

• розбиття множини даних на кластери - набір розбиття множина зразків даних на класи еквівалентності на шкалі відношення а-квазіеквівалентності.

Стандарти зачіпають три основних аспекти Data Mining. По-перше, уніфікацію інтерфейсів, за допомогою яких будь-який додаток може добути доступ до функціональності Data Mining. Тут склалося два направлення. Це стандартизація інтерфейсів для об'єктних мов програмування (CWM Data Mining, JDM, OLE DB for Data Mining) і спроби розробки надбудови для мови SQL, яка дозволяла б звертатися до інструментарію Data Mining,

вбудованому безпосередньо в реляційну базу даних (SQL / MM, OLE DB for Data Mining).

Другий аспект стандартизації – це вироблення єдиної угоди по зберіганні і передачі моделей Data Mining. Неважко здогадатися, що основою для подібного стандарту є мова XML. Сам стандарт носить назву PMML (Predicted Model Markup Language). І нарешті, існує стандарт CRISP, який дає рекомендації з організації процесу Data Mining в цілому[20].

## 2.2 Інструменти реалізації

### 2.2.1 Обґрунтування використання таблиці Excel

Microsoft Excel (також іноді називається Microsoft Office Excel) — програма для роботи з електронними таблицями, створена корпорацією Microsoft для Microsoft Windows, Windows NT і Mac OS, а також Android, iOS і Windows Phone. Вона надає можливості економіко-статистичних розрахунків, графічні інструменти і, за винятком Excel 2008 під Mac OS X, мова макропрограммирования VBA (Visual Basic for Application). Microsoft Excel входить до складу Microsoft Office і на сьогоднішній день Excel є одним з найбільш популярних програм у світі.

Процес інтелектуального аналізу даних можна розбити на етапи:

1. Розуміння та формулювання задачі аналізу.
2. Підготовка даних до автоматичного аналізу.
3. Застосування методів Data Mining.
4. Перевірка створених моделей.
5. Інтерпретація моделей людиною.

На другому етапі аналізу даних, не має значення звідки та які дані будуть братися, головним моментом у обробці даних – є упорядкованість. Дані повинні бути заздалегідь скомпоновані у одну таблицю, яка і буде підлягати обробці. Тобто потрібно привести дані до форми яка дозволить зробити аналіз.

Іншими словами, необхідно виробити якийсь чіткий набір числових або нечислових параметрів, що характеризують лист. Ця задача найменш автоматизована в тому сенсі, що вибір системи параметрів виробляється людиною, хоча, звичайно, їх значення можуть обчислюватися автоматично. Після вибору описують параметрів вивчаються дані можуть бути представлені у вигляді прямокутної таблиці, де кожен рядок являє собою окремий випадок, об'єкт або стан досліджуваного об'єкта, а кожна колонка— параметри, властивості або ознаки всіх досліджуваних об'єктів. Більшість методів Data Mining працюють тільки з подібними прямокутними таблицями.

Отримана прямокутна таблиця поки ще є занадто сирим матеріалом для застосування методів Data Mining, і вхідні в неї дані необхідно попередньо обробити. По-перше, таблиця може містити параметри, що мають однакові значення для всієї колонки. Якщо б досліджувані об'єкти характеризувалися тільки такими ознаками, вони були б абсолютно ідентичні, значить, ці ознаки ніяк не індивідуалізують досліджувані об'єкти. Отже, їх треба виключити з аналізу. По - друге, таблиця може містити певний категоріальний ознака, значення якого в усіх записах різні. Ясно, що ми ніяк не можемо використовувати це поле для аналізу даних і його треба виключити. Нарешті, просто цих полів може бути дуже багато, і якщо всі їх включити в дослідження, то це значно збільшить час обчислень, оскільки практично для всіх методів Data Mining характерна сильна залежність часу від кількості параметрів (не менш ніж квадратична, а нерідко і експоненціальна). В той же час залежність часу від кількості досліджуваних об'єктів лінійний або близька до лінійної. Тому в якості попередньої обробки даних необхідно, по-перше, виділити безліч ознак, які найбільш важливі в контексті даного дослідження, відкинути явно незастосовні з-за константності або надмірної варіабельності і виділити ті, які найбільш ймовірно увійдуть в шукану залежність. Для цього, як правило, використовуються статистичні методи, засновані на застосуванні кореляційного аналізу, лінійних регресій і т. д. Такі методи дозволяють швидко, хоч і приблизно, оцінити вплив одного параметра на інший.

Ми обговорили очищення даних у стовпчиках таблиці (ознаками). Точно також буває необхідно провести попередню очистку даних по рядках таблиці (записів). Будь-яка реальна база даних зазвичай містить помилки, дуже неточно визначені значення, записи, що відповідають якимось рідкісних, виняткових ситуацій, та інші дефекти, які можуть різко знизити ефективність методів Data Mining, що застосовуються на наступних етапах аналізу. Такі записи необхідно відкинути. Навіть якщо подібні "викиди" не є помилками, а являють собою рідкісні виняткові ситуації, вони все одно

навряд чи можуть бути використані, оскільки по декількох точках статистично значимо судити про шуканої залежно неможливо. Ця попередня обробка або препроцессинг даних і складає другий етап.

### 2.2.2 Вибір програмного середовища

Microsoft Visual Studio — лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як у рідному, так і в керованих кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight [2].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Інші вбудовані інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу програми, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення [5].

Visual Studio 2010 випустили у 2010 році разом з .NET Framework 4.0. Visual Studio має підтримку мови програмування C++, яка використовувалась у програмному коді в процесі виконання дипломної роботи, Visual Basic.NET 10.0, а також мову F#, відсутніх у попередніх версіях.

Програма написана в середовищі Microsoft Visual Studio 2013 — бо вона є необхідним засобом для незалежних розробників, що дозволяє вирішувати основні завдання розробки. Система спрощує створення, налагодження та розгортання додатків на різних платформах, включає в себе широкий спектр функцій, які дозволяють розробникам виконувати побудова, налагодження, тестування модулів і розгортання високоякісних додатків на різних платформах, включаючи Windows, веб-середовище, Office і багато інших.

### 2.2.3 Вибір і обґрунтування мови програмування

Для реалізації поставлених задач було обрано мову програмування C++.

C ++ (читається сі-плюс-плюс) – компільований, статично типізований мова програмування загального призначення [2].

Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає в себе поширені контейнери і алгоритми, введення-виведення, регулярні вирази, підтримку багатопоточності і інші можливості. C ++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником - мовою C, - найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування.

C++ широко використовується для розробки програмного забезпечення, будучи одним з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також розважальних програм (ігор). Існує безліч реалізацій мови C ++, як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C ++, Intel C ++

Compiler, Embarcadero (Borland) C ++ Builder і інші. C ++ зробив величезний вплив на інші мови програмування, в першу чергу на Java і C # [2].

Синтаксис C ++ успадкований від мови C. Одним з принципів розробки було збереження сумісності з C. Проте, C ++ не є в строгому сенсі надбезліччю C; безліч програм, які можуть однаково успішно транслюватися як компіляторами C, так і компіляторами C ++, досить велике, але не включає всі можливі програми на C.

З одного боку, C ++ є нащадком Симула, яку Алан Кей визначив як «Алгол з класами», і тому буде актуальною оцінка C ++ в порівнянні з іншими мовами з сімейства нащадків Алгола (Basic, Pascal, Java, C #, Visual Basic, Delphi, D, Oberon і ін.). З іншого боку, C ++ претендує на мультипарадигменість і універсальну застосовність (на відміну від Ci, орієнтованого на дуже вузьке коло завдань), і використовується в промисловості набагато ширше інших нащадків Алгола, і тому буде актуальною оцінка C ++ в порівнянні з усім різноманіттям вживаних мов, включаючи і Ci. Щоб уникнути повторень, оцінки зазвичай поєднуються.

C ++ - мову, що складається еволюційно. На відміну від мов з формальним визначенням семантики, кожен елемент C ++ запозичувався з інших мов окремо і незалежно від інших елементів (ніщо з запропонованого C ++ за всю історію його розвитку не було нововведенням в Computer Science), що зробило мову надзвичайно складним, з безліччю дублюються і взаємно суперечливих елементів, блоки яких засновані на різних формальних базах. В цьому відношенні C ++ повторює шлях PL / 1, але, на відміну від останнього, тривалий повсюдне використання C ++ забезпечив вибір мови Ci в якості відправної точки.

Критики C ++ протиставляють йому який-небудь конкретний мову, а навпаки, стверджують, що для будь-якого випадку застосування C ++ завжди існує альтернативний інструментарій, що дозволяє вирішити ту ж задачу більш ефективно і якісно. У свою чергу, прихильники C ++ вважають некоректним порівнювати різні аспекти C ++ з абсолютно різними мовами,

так як загальний набір засобів і можливостей C ++ істотно ширше, ніж в більшості мов, з якими проводиться порівняння, і сама по собі широта можливостей, на їх погляд, є вагомим виправданням недосконалості кожної окремо взятої можливості.

Слід зазначити, що розгляд елементів мови окремо означає відмову від розгляду мови як системи, а отже, і від очікування синергізму, - що робить обмеженим (кінцевим) потенціал зростання складності абстракцій, які можна виразити за допомогою цієї мови, і відповідно звужує спектр реалізованих на ньому програмних систем.

C ++ заявляється як багатоплатформовий: стандарт мови накладає мінімальні вимоги на ЕОМ для запуску двійкові програми. На практиці, для написання портіруемость коду на C ++ потрібна величезна майстерність і досвід, і «недбалі» коди на C ++ з високою ймовірністю можуть виявитися непортуємими. Тонке володіння C ++ в принципі може зробити код на C ++ настільки ж портуємість, що і код на Сі (хоча, на думку Лінуса Торвальдса, C ++ при цьому фактично скоротиться до свого підмножини Сі). Однак, критики C ++ стверджують, що вивчення і використання одночасно всіх мов, що протиставляються C ++ (що не викликають серйозних проблем при портуванні), в сумі вимагає приблизно тих же інтелекту, зусиль і тимчасових витрат, що і вивчення і використання одного тільки C ++ на висококласному рівні - в зв'язку з чим стає актуальною також оцінка порогу входження і результативності (продуктивності і якості праці програмістів).

C ++ містить засоби розробки програм контрольованої ефективності для широкого спектра задач, від низькорівневих утиліт і драйверів до вельми складних програмних комплексів. Зокрема:

Висока сумісність з мовою Сі: код на Сі може бути з мінімальними переробками скомпільовано компілятором C ++. Зівнішньоязиковий інтерфейс є прозорим, так що бібліотеки на Сі можуть викликатися з C ++ без додаткових витрат, і більш того - при певних обмеженнях код на C ++

може експортуватися зовні не відмінних від коду на Сі (конструкція extern «C»).

Як наслідок попереднього пункту - обчислювальна продуктивність. Мова спроектований так, щоб дати програмісту максимальний контроль над усіма аспектами структури і порядку виконання програми. Один з базових принципів C ++ - "не платиш за те, що не використовуєш» (див. Філософія C ++) - тобто жодна з мовних можливостей, що призводить до додаткових накладних витрат, не є обов'язковою для використання. Є можливість роботи з пам'яттю на низькому рівні.

Підтримка різних стилів програмування: традиційне імперативне програмування (структурний, об'єктно-орієнтоване), узагальнене програмування, функціональне програмування, що породжує метапрограмування.

Автоматичний виклик деструкторів об'єктів в адекватному порядку (зворотному виклику конструкторів) спрощує і підвищує надійність управління пам'яттю і іншими ресурсами (відкритими файлами, мережевими з'єднаннями, сполуками з базами даних і т. П.).

Перевантаження операторів дозволяє коротко і ємко записувати вирази над користувацькими типами у природному алгебраїчній формі.

Є можливість управління константністю об'єктів (модифікатори const, mutable, volatile). Використання константних об'єктів підвищує надійність і служить підказкою для оптимізації. Перевантаження функцій-членів за ознакою константності дозволяє визначати вибір методу в залежності мету виклику (константний для читання, неконстантний для зміни). Оголошення mutable дозволяє зберігати логічну константність побачивши ззовні коду, що використовує кеші і ледачі обчислення. Шаблони C ++ дають можливість побудови узагальнених контейнерів і алгоритмів для різних типів даних. Попутно шаблони дають можливість виробляти обчислення на етапі компіляції.

Можливість вбудовування предметно-орієнтованих мов програмування в основний код. Такий підхід використовує, наприклад бібліотека Boost.Spirit, що дозволяє задавати EBNF-граматику парсерів прямо в коді C + +. Boost.Spirit реалізує рекурсивно-спадний алгоритм, що накладає відповідні обмеження (такі як неприпустимість лівої рекурсії).

Доступність. Для C ++ існує величезна кількість навчальної літератури, перекладеної на всілякі мови. Мова має високий поріг входження, але серед всіх мов такого роду має найбільш широкі можливості.

# РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ

## 3.1 Вирішення поставленої задачі

### 3.1.1 Постановка задачі

У задачі кластеризації віднесення кожного з об'єктів даних здійснюється до одного (або кількох) з заздалегідь невизначених класів. Розбиття об'єктів даних по кластерам здійснюється при одночасному їх формуванні. Визначення кластерів і розбиття по ним об'єктів даних виражається в підсумковій моделі даних, яка є рішенням задачі кластеризації.

Зважаючи на особливе становище завдання кластеризації в списку завдань інтелектуального аналізу даних було розроблено безліч способів її рішення. Один з них – побудова набору характеристичних функцій класів, котрі показують, відноситься об'єкт даних до даного класу чи ні. Характеристична функція класу може бути двох типів:

1) дискретна функція, приймаюча одне з двох певних значень, зміст яких в належності / неналежності об'єкта даних заданому класу;

2) функція, приймаюча речові значення, наприклад з інтервалу 0 ... 1. Чим ближче значення функції до одиниці, тим більше об'єкт даних належить заданому класу.

Загальний підхід до вирішення завдання кластеризації став можливий після розвитку Л. Заде теорії нечітких множин. В рамках даного підходу вдається формалізувати якісні поняття, невизначеність, притаманну реальним даними і процесам. Успіх цього підходу пояснюється ще й тим, що в процесі аналізу даних бере участь людина, оцінки і судження якого розпливчасті і суб'єктивні. Доречно навести висловлювання Л. Заде, основоположника теорії нечітких множин: «... потрібна нова точка зору, новий комплекс понять і методів, в яких нечіткість приймається як універсальна реальність людського існування».

Застосовуючи теорію нечітких множин для вирішення завдання кластеризації, можливі різні варіанти введення нечіткості в методи, які

вирішують це завдання. Нечіткість може враховуватися як в поданні даних, так і при описі їх взаємозв'язку. Крім того, дані можуть як мати, так і не мати кількісної природою. Проте в багатьох практичних задачах дані, які необхідно досліджувати, є результатом накопиченого досвіду в тій чи іншій сфері людської діяльності і часто мають кількісне уявлення. Облік нечіткості самих досліджуваних даних, в загальному випадку, - серйозна проблема. Тому як в існуючих алгоритмах, так і в підході, запропонованому в даному виданні, не робиться ніяких припущень про нечіткість самих вихідних даних. Вважається, що дані є чіткими і виражені кількісно.

Описувати нечіткі взаємозв'язку даних можна різними способами. Одним з таких способів, які знайшли широке поширення в використовуваних в даний час алгоритмах нечіткої кластеризації даних, є опис взаємозв'язку даних через їхнє ставлення до деяких еталонним зразком – центрам кластерів. В даних алгоритмах нечіткість проявляється в описі кластерів як нечітких множин, що мають ядро в центрі кластера. З іншого боку, взаємозв'язок даних в умовах невизначеності можна враховувати за допомогою апарату нечітких відносин між окремими зразками даних, не вдаючись при цьому до поняття центру кластера. Такий підхід не знайшов ще широкого поширення на практиці, хоча, очевидно, є більш універсальним. В даному виданні робиться спроба заповнити цю прогалину і показати ті переваги, які дає використання зазначеного поняття для завдання кластеризації. Отже, перейдемо до постановці завдання кластеризації.

Дано – набір даних з наступними властивостями:

• кожен екземпляр даних виражається чітким числовим значенням;

• клас для кожного конкретного екземпляра даних невідомий. знайти:

• спосіб порівняння даних між собою (міру подібності);

• спосіб кластеризації;

• розбиття даних по кластерам.

Формально задача кластеризації описується наступним чином.

Використовувати дані інтернет-магазина з продажу товарів, які є безліччю об'єктів даних. Кожен об'єкт представлений набором атрибутів. Потрібно побудувати безліч кластерів C і відображення F безлічі 1 на безліч C, т. Е. F: l-> C. Відображення F задає модель даних, що є вирішенням завдання. Якість рішення задачі визначається кількістю вірно класифікованих об'єктів даних. Результат кластеризації відобразити у вигляді діаграми, вихідними даними повинні задаватися на формі, а аналізовані дані повинні розташовуватися в Excel документі в придатному для аналізу вигляді.

Таблиця, побудована у Microsoft Excel, може складатися з різних наборів даних, головне щоб усі поля були заповнені. Кожна строчка є набором параметрів – вхідній вектор. Таблиця має складатися з більш ніж двох стовпців і великого набору рядків.

### 3.1.2 Опис алгоритму кластеризації даних

На сьогоднішній день сучасні обчислювальні машини і комп'ютерні мережі дозволяють накопичувати великі масиви інформації для задач обробки та аналізу. На жаль, сама по собі машинна форма подання даних містить інформацію, необхідну людині, в прихованому вигляді, і для її вивчення потрібно використовувати спеціальні методи аналізу даних [5].

Величезний обсяг інформації дозволяє отримати більш точні розрахунки і аналіз, однак, він перетворює пошук інформації в складну задачу.

Відомий спосіб аналізу даних - Кластеризація, може застосовуватися в багатьох областях, де необхідно дослідження експериментальних або статистичних даних.

Аналітику часто легше виділити групи схожих об'єктів, вивчити їх особливості і побудувати для кожної групи окрему модель, ніж створювати одну загальну модель для всіх даних [11]. Таким прийомом постійно

користуються в маркетингу, виділяючи групи клієнтів, покупців, товарів і розробляючи для кожної з них окрему стратегію [11].

Основним завданням кластеризації є пошук незалежних груп (кластерів) у всій безлічі аналізованих даних. Кластерний аналіз дозволяє краще зрозуміти дані. Так само, угруповання подібних об'єктів дозволяє скорочувати їх число, що призводить до спрощення аналізу. Кожна група буде містити «подібні» об'єкти, а об'єкти різних груп повинні максимально відрізнятися. Перелік груп чітко не заданий і визначається в процесі роботи алгоритму [6].

Застосування кластерного аналізу зводиться до наступних етапів:

1. Вибір масиву даних для кластеризації.

2. Визначення головних (центральних) змінних для оцінки об'єктів у вибірці.

3. Визначити подібність між усіма об'єктами щодо центральних змінних.

4. Використання кластерного аналізу для створення груп подібних об'єктів (кластерів).

5. Висновок результатів аналізу.

Отриманий результати можна відкоригувати, вибравши інший метод аналізи для отримання максимально точних результатів.

K-means - простий повторюваний алгоритм кластеризації, який розбиває великий набір даних на групи відповідно до заданого числа кластерів k. Алгоритм досить просто реалізувати і програмувати, є відносно швидким, підходить під аналіз практично будь-яких даних. Часто використовується для аналізу закономірностей в покупках. Алгоритм k-means історично один з найважливіших алгоритмів інтелектуального аналізу даних.

Історично склалося так, що k-means був відкритий декількома дослідниками різних дисциплін, в першу чергу Ллойдом (1957, 1982), Фордж (1965), Фрідманом і Рубіном (1967), і МакКуїном (1967) [10].

Алгоритм k-means застосовується масиву значень точок в d-вимірному векторному просторі. Таким чином, це кластери набору d-мірних векторів,, де позначає j-ий об'єкт або «точку даних». Як вже говорилося, k-means є алгоритмом кластеризації, який розділяє D на k кластерів точок. Тобто, алгоритм k-means об'єднує всі точки даних в D так, що кожна точка потрапляє в один і тільки один з k кластерів. Можна відстежити, яка точка знаходиться в якому кластері, призначивши кожній точці номер кластера. Точки з таким же номером кластера знаходяться в одному і тому ж кластері, в той час як точки з різними номерами кластера знаходяться в різних кластерах [10].

Алгоритм діє за принципом мінімізації сумарного квадратичного відхилення точок кластерів від центрів цих кластерів (Центроїд - точок, які є центрами кластерів). В алгоритмах кластеризації угруповання точок відбувається підбором подібних самим собі, схожим за більшістю ознак. Алгоритм k-means використовує міру близькості - Евклідова відстань.

$$V=\sum_{i=1}^{k}\sum_{x_j\in S_i}(x_j-\mu_i)^2 \quad , \qquad (1)$$

де $\kappa$ – число кластерів, $x_j$ – кожний вектор представлений крапкою, $i = 1,2,\ldots,k$ , $S_i$ – отримані кластери и $\mu_i$ – центри мас векторів $x_j$ (центроіди).

Значення k є основним з вхідних даних алгоритму і задається дослідником кожен раз вручну або випадковим чином [9].

Алгоритм k-means призводить до мінімізації підсумкового квадрата Евклидова відстані між кожним вектором $x_j$ і подібною точкою кластера $\mu_i$ . Рівняння (1) є цільовою функцією методу k-means.

Алгоритм k-means:

1. Випадковим чином вибирається к-точок (центроїди) з початкового безлічі точок.

2. Використовуючи формулу (1) розподіляємо точки по кластерам, щодо Центроїд.

3. Знаходимо нове положення центроїдів, обчисливши центр кожного кластера.

4. Виконуємо пункт 2 і 3 до тих пір, поки центроїди не перестануть змінювати своє положення або до певного порогу зміни положення центроїдів.

Кожна ітерація потребує N * k порівнянь, що ясно видно з алгоритму (рис. 2), що визначає складність однієї ітерації.

Число ітерацій може залежати від N і ітерації змінюються в залежності від N. А це означає, що чим більше точок в безлічі (N), тим довше буде працювати алгоритм.

Для скорочення часу роботи алгоритму використовують розпаралелювання етапів розподілу точок по кластерам.

Кожен процес можна розбити на стільки ж потоків і тоді початкова множина даних розбивається на таку ж кількість частин, при цьому кожен потік буде мати справу зі своїм обсягом даних, незалежно він інших.

При аналізі великих обсягів даних процес зміни положення центроїдів може займати багато часу і при цьому центроїди відхиляються на незначні відстані, тому заздалегідь встановлюють поріг відхилення Центроїд (крайня точка, до якої ведеться ітерація) щодо попереднього положення. Це дозволяє скоротити час для розбиття масиву даних на групи.

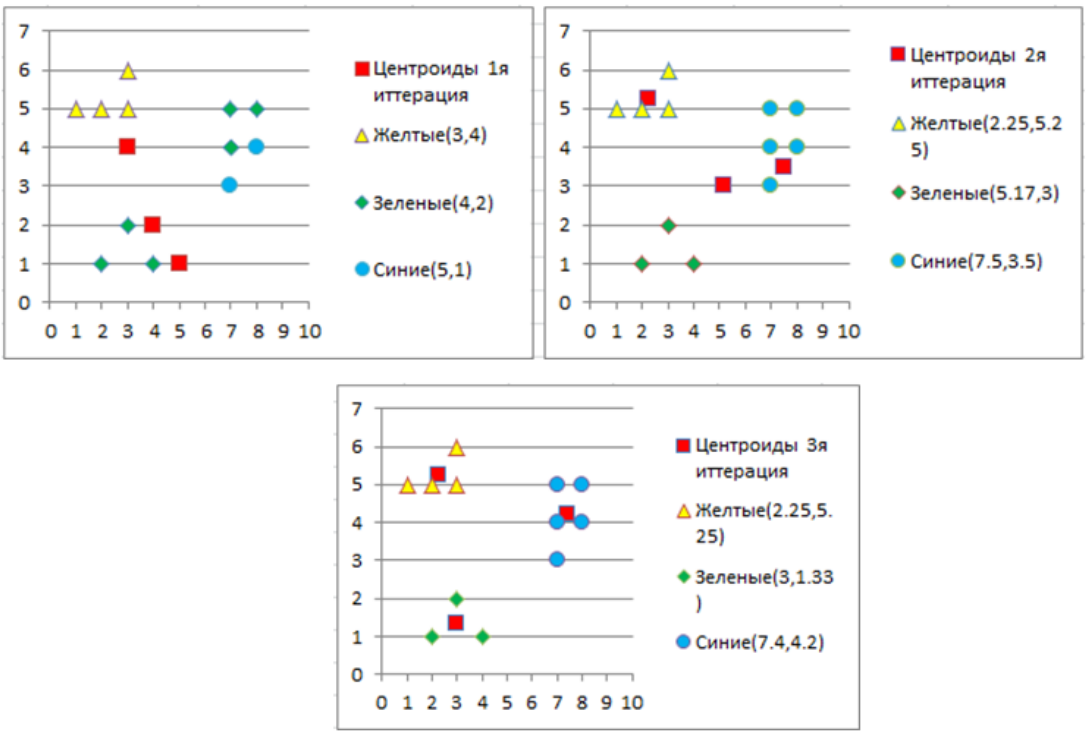


Рис.3.1. Результат кластеризації методом k-середніх

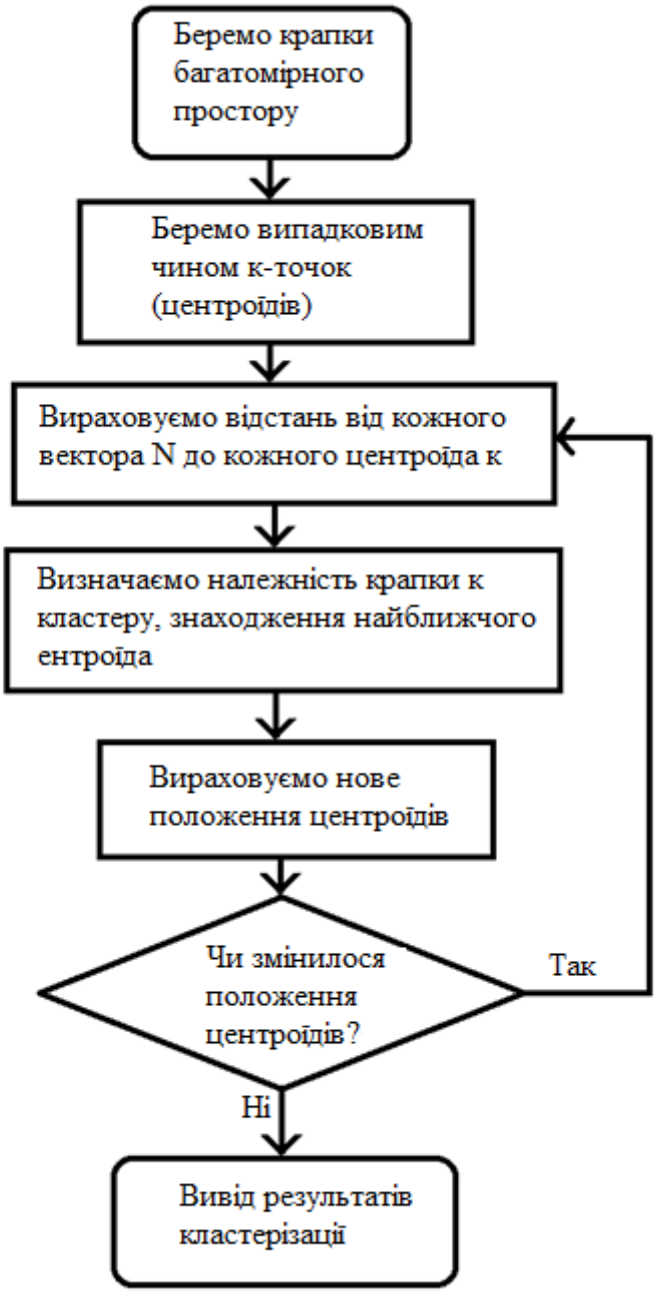


Рис.3.2. Схема алгоритму *k*-середніх

Після розбиття масиву даних на групи використовуються інші методи Data Mining, для подальшого аналізу і з'ясування причин такого розбиття.

## 3.2 Створення клієнтської частини проекту

### 3.2.1 Створення інтерфейсу користувача

Якість будь-якої програми багато в чому визначається якістю розробки інтерфейсу «ОПЕРАТОР-ЕОМ». Призначений для користувача програмне забезпечення повинне забезпечувати повний комплекс засобів для взаємодії як з базою даних, так і з користувачем, як те: занесення, зміна і видалення даних, моніторинг і діагностика роботи інформаційної системи, здобуття вибірок інформації і звітів у вигляді придатному як для перегляду на моніторі, так і для друку на принтері.

Інерфейс даної роботи створювався за допомогою GUI – графічного інтерфейсу користувача, різновиду інтерфейсу користувача вбудованого у платформу Visual Studio 2013, що дозволяє швидко та зручно розробляти візуальні об’єкти для швидкого програмування певних функцій на виконання поставлених завдань.

За допомогою GUI користувач має можливість бачити готові об’єкти. Які можна розміщувати на формі та подвійним кліком миші формувати вже готовий код функції при натисканні на цей об’єкт.

Графічний інтерфейс користувача є частиною інтерфейсу користувача, який формулює взаємодію користувача з програмою на рівні візуалізації.

Графічний інтерфейс є дуже дружелюбним та зручним у використанні та він є єдиним способом візуалізації в обробці графіки.

Для створення інтерфейсу даної роботи використовувались такі графічні об’єкти як Button, ProgressBar, Chart, TextBox, GroupBox, Lable, MenuStrip.

MenuStrip – розташовано у верхній частині вікна, меню має вкладки, які дозволяють переміщуватись по об’єктам за призначенням.

Перша вкладка «File», в ній є випадний список з полями «Save as…» - зберігає знайдені результати аналізу даних у заданому форматі. «Edition» -

Наступне поле вкладки «Exit» - виходить з програми, попередньо запитуючи у користувача чи дійсно він хоче вийти.

Вкладка «Help» - має інформативний характер відносно даної системи.

Button «File selection...» – ця кнопка призначена для вибору документу на диску в форматі Microsoft Excel для подальшого аналізу даних.

Button «Processing» - кнопка призначена для запуску алгоритму аналізу даних.

GroupBox – комбінує об’єкти системи у єдине поле для зручності.

Lable – підпис або назва певних полів, для іменування об’єктів графічного інтерфейсу.

TextBox – поля вводу даних для заповнення кількості кластерів та при необхідності дати межу для завершення роботи алгоритму.

Chart - об’єкт класу який відображає дані у вигляді графіків. У даному випадку він аналізує дані та виводить отриману інформацію і графічному вигляді. Використовується в якості кореневого елемента керування Chart.

ProgressBar – об’єкт класу який відображає рівень завершеності обробки даних після натиснення на кнопку «Processing».

OpenFileDialog – клас який дозволяє відображати діалогове вікно, яке в свою чергу дозволяє відкривати файли. Цей клас дозволяє перевірити, чи існує файл і відкрити його. ShowReadOnly Властивість визначає, чи відображається в діалоговому вікні прапорець тільки для читання. ReadOnlyChecked вказує, чи встановлений прапорець «тільки для читання».

SaveFileDialog – клас, який запитує у користувача місцезнаходження для збереження файлу.

Об’єкти класів OpenFileDialog, SaveFileDialog використовуються для вибору документа з даними які треба проаналізувати та зберегти інформацію після обробки.

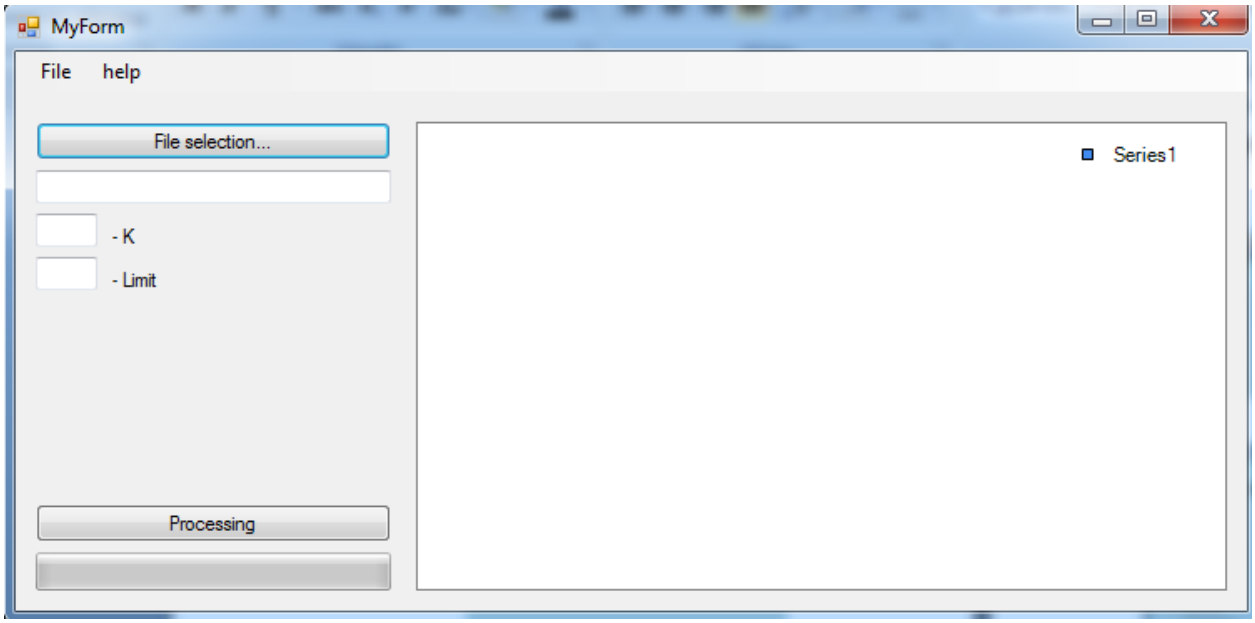


Рис.3.3 Вікно програми

### 3.2.2 Програмна реалізація проекту

В процесі реалізації проекту використовувались ряд бібліотек:

```
#include <iostream>
#include <conio.h>
#include <fstream>
#include <vector>
#include <math.h>
```

Бібліотека iostream – об'єктно-орієнтована ієрархія класів, де використовується і множинне, і віртуальне успадкування. У ній реалізована підтримка для файлового введення / виводу даних вбудованих типів. Крім того, розробники класів можуть розширювати цю бібліотеку для читання і запису нових типів даних.

Conio.h (від англ. Console input-output - консольне введення-виведення) – заголовки, використовуваний в старих компіляторах, що працюють в операційних системах MS-DOS, для створення текстового інтерфейсу користувача. Проте, він не є частиною мови програмування Сі, стандартної бібліотеки мови Сі, ISO C або необхідної стандартом POSIX.

Цей заголовки оголошує кілька бібліотечних функцій для роботи з «консольним введенням і висновком» програми. Більшість компіляторів мови Сі, призначених для DOS, Windows 3.x, Phar Lap, DOSX, OS / 2 або Win32 мали цей файл і забезпечували супутні бібліотечні функції в бібліотеці Сі за замовчуванням. Більшість компіляторів мови Сі, призначених для UNIX і Linux, не мають цього файлу і не забезпечують супутніх бібліотечних функцій.

Fstream – визначає кілька класів, що підтримують операції iostreams для послідовностей, що зберігаються в зовнішніх файлах.

Vector – визначає вектор класів шаблонів контейнерів і деякі допоміжні шаблони.

Vector – це контейнер, який впорядковує елементи даного типу у вигляді лінійної послідовності. Він забезпечує швидкий довільний доступ до

будь-якого елементу і дозволяє динамічно додавати елементи в послідовність і видаляти їх. vector є найбільш підходящим типом контейнера для послідовності, коли на першому місці стоїть продуктивність довільного доступу.

Math.h – надає  константи і статичні методи для тригонометричних, логарифмічних та інших спільних математичних функцій.

Та простір імен:

```
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Collections::Generic;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Drawing::Drawing2D;
using namespace System::IO;
using namespace System::Windows::Forms::DataVisualization::Charting;
using namespace std;
```

System – простір імен, що мають фундаментальні та базові класи, які визначають найчастіше використовувані типи значень та значень посилань, події і обробники подій, інтерфейси, атрибути і виключення обробки.

System.ComponentModel Простір імен надає класи, використовувані для реалізації поведінки компонентів і елементів управління під час виконання і під час розробки. Це простір імен містить базові класи і інтерфейси для реалізації атрибутів і перетворювачів типів, прив'язки до джерел даних і ліцензування компонентів.

Простору імен System.Collections містять типи, що визначають різні стандартні, спеціальні і універсальні об'єкти колекцій.

Простір імен System.Collections.Generic містить інтерфейси і класи, що визначають універсальні колекції, які дозволяють користувачам створювати

строго типізовані колекції, що забезпечують підвищену продуктивність і безпеку типів у порівнянні з неуніверсальними строго типізований колекціями.

System.Windows.Forms простір імен містить класи для створення додатків Windows, що користуються перевагами повного призначеного для користувача інтерфейсу, що надаються в операційній системі Microsoft Windows.

Простір імен System.Data містить класи для доступу до даних з різних джерел і для управління цими даними. Простір імен верхнього рівня і кілька дочірніх просторів імен утворюють архітектуру ADO.NET і постачальників даних ADO.NET. Наприклад, доступні постачальники для SQL Server, Oracle , ODBC і OleDB. Інші дочірні простору імен містять класи, використовувані моделлю EDM ADO.NET і службами даних WCF.

Батьківське простір імен System.Drawing містить типи, що підтримують базові графічні функції GDI +. Дочірні простору імен підтримують більш складні функції двомірної і векторної графіки, додаткові функції обробки зображень, а також служби, пов'язані з друком і типографікою. Дочірнє простір імен також містить типи, які розширюють логічні і графічні можливості призначеного для користувача інтерфейсу під час розробки.

System.Drawing.Drawing2D Простір імен надає складні функції двомірної і векторної графіки.

Простору імен System.IO містять типи, що підтримують введення і виведення, включаючи можливості читання і запису даних в потоках як синхронно, так і асинхронно, стиснення даних в потоках, створення і використання ізольованих сховищ, зіставлення файлів логічним адресним просторам додатків, зберігання різних об'єктів даних в одному контейнері, взаємодії з використанням анонімних або іменованих каналів, реалізації призначеного для користувача ведення журналу та обробки вхідних і вихідних потоків даних в послідовних портах.

System.Windows.Forms.DataVisualization.Charting Простір імен містить методи і властивості для елементу управління діаграми Windows forms.

### 3.3 Інструкція з використання

Для запуску програми необхідно запустити exe файл. Користувачу відкриється вікно на якому будуть розташовані дві кнопки, поля для заповнення та поле діаграми.

У верхній частині програми знаходиться меню, де можна побачити такі вкладки: «File», «Help».

Вкладка «File» містить випадаючий список, який містить такі поля: «Save as…», «Exit».

«Save as…» - дозволяє зберігати отримані результати у заданому форматі.

«Exit» - дозволяє вийти з програми.

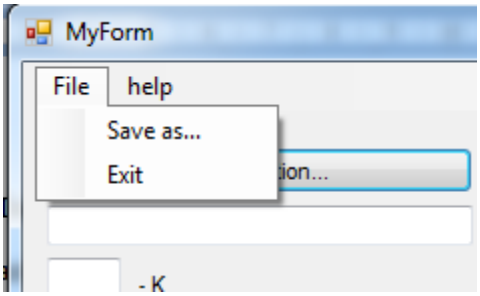


Рис.3.4. Вкладка «File»

Вкладка «Help» містить випадаючий список з полем «About program», в якому у стислій формі можна побачити інформацію о програмі.

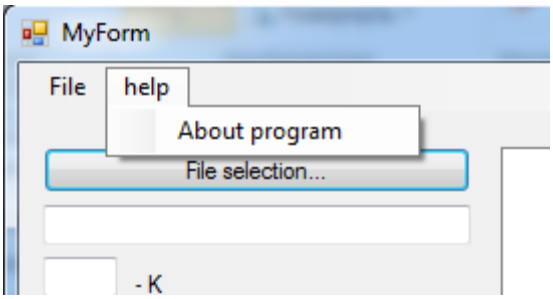


Рис.3.5. Вкладка «Help»

Для початку обробки підготовленого файлу Excel необхідно натиснути на кнопку «File selection...», де відкриється нове вікно, в якому треба буде вибрати необхідний файл для обробки.

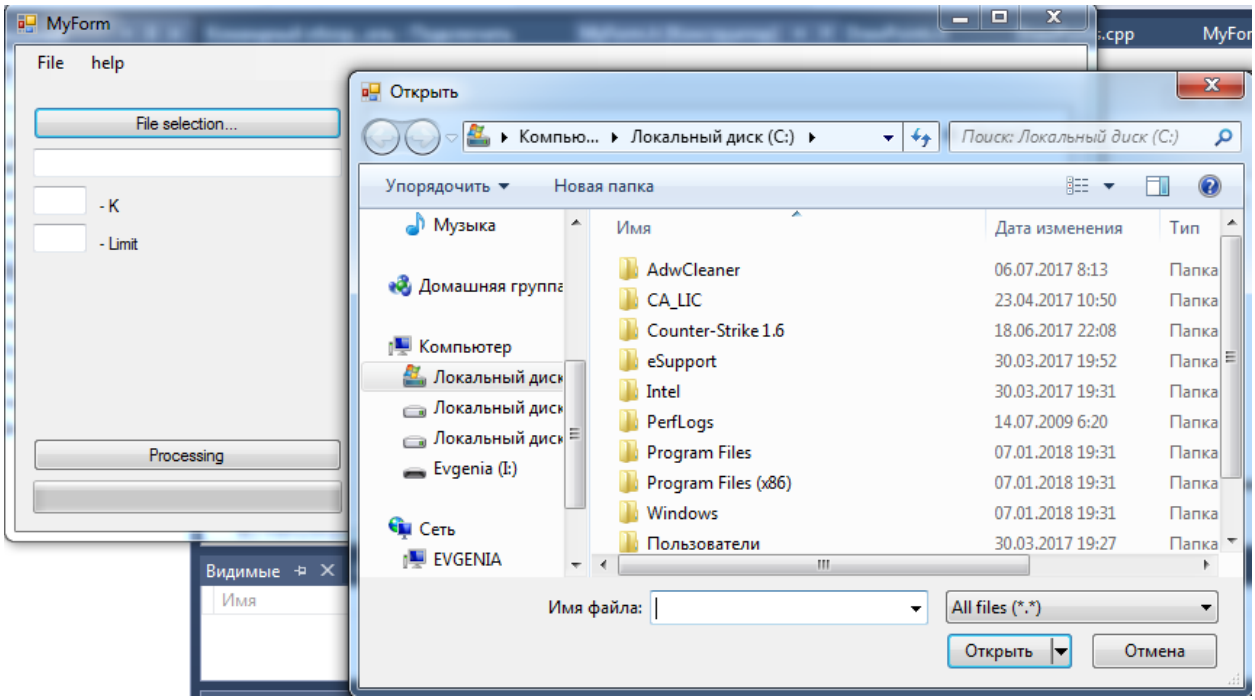


Рис.3.6. Вікно вибору файлу

Після вибору файлу, місце його знаходження відобразиться у полі нижче кнопки. Перед запуском програми треба указати кількість розбиття у полі поряд з написом «К», що означає «Кластери». Також можна зазначити обмеження обробки аби уникнути зайвих витрат часу на обробку, для цього потрібно в полі «Limit» указати значення різниці між попереднім положенням центроідів та поточним. Таким чином програма буде працювати до тих пір поки не досягне цього значення.

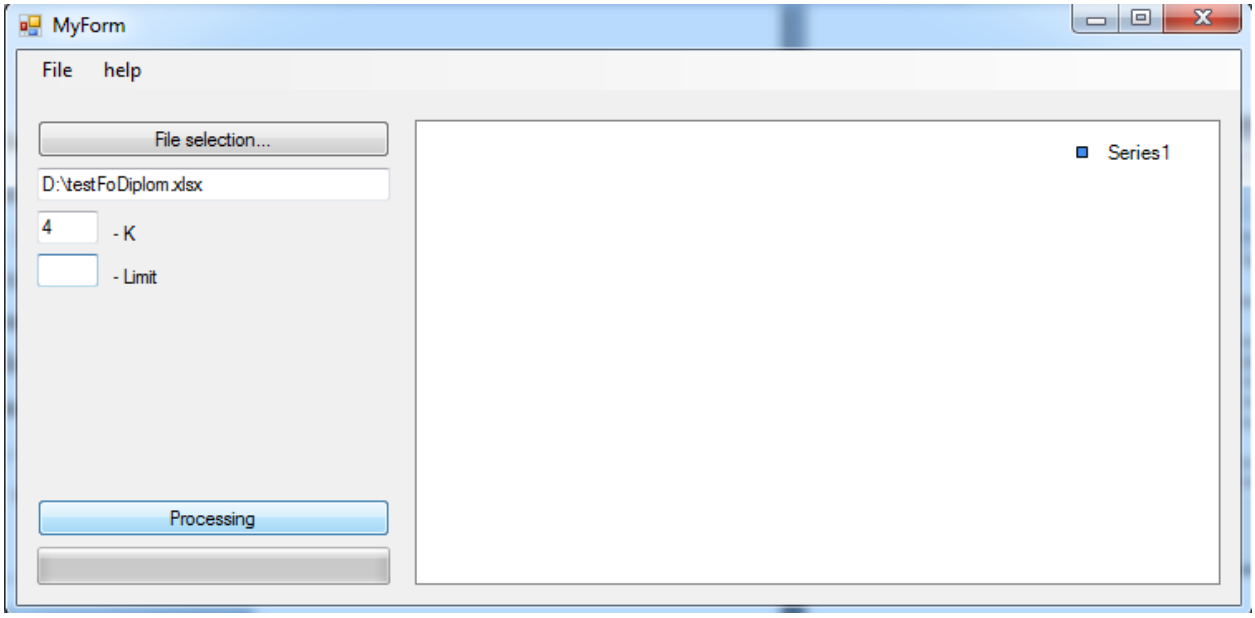


Рис.3.7. Заповнене вікно програми

Після заповнення усіх полів, треба натиснути на кнопку «Processing». Програма почне виконувати алгоритм та відобразить результати у графічному вигляді праворуч вікна.

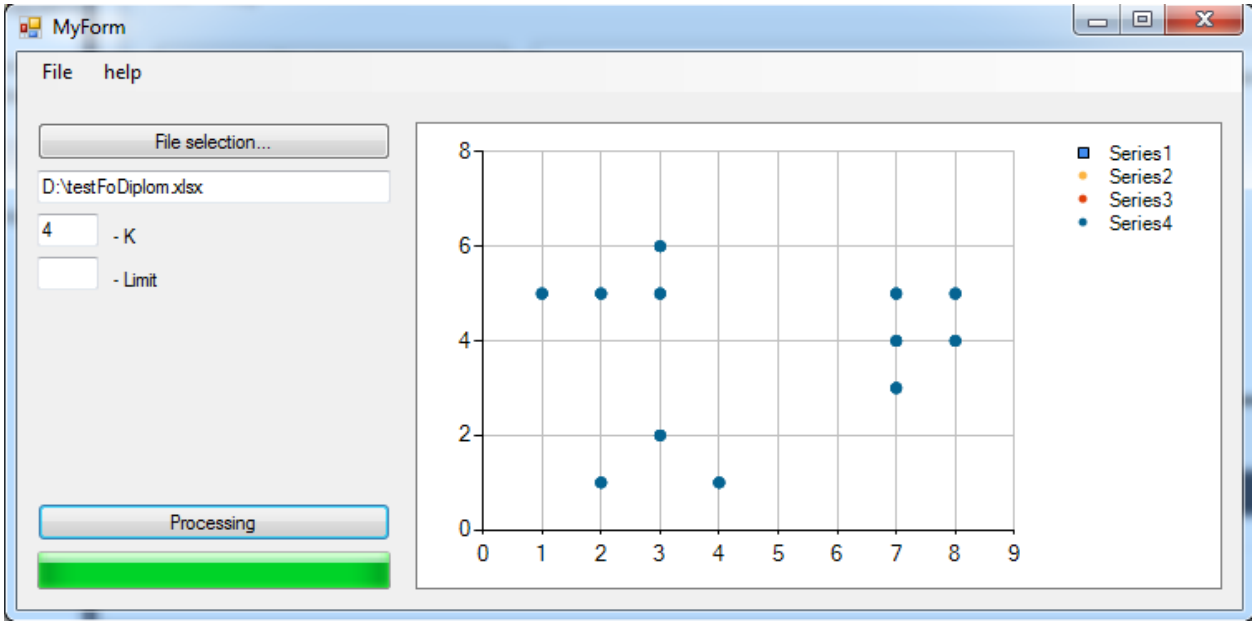


Рис.3.8. Процес обробки даних програмою

# ВИСНОВКИ

Розроблена програма аналізу даних інтернет магазина, яка дозволяє вибрати документ для обробки та методом к-середніх зробити аналіз завантажених даних.

Програма написана на мові C++ в середовищі Microsoft Visual Studio 2013, що володіє розвиненими функціональними можливостями, ідеально підходить для навчання, розробки і нарощування функціональних додатків. Дані для обробки беруться з документу Excel та завантажуються завдяки класу OpenFileDialog.

Засобами GUI були створені компоненти управління системою.

Результати обробки даних відоброжаються у вигляді діаграми, яку можна зберегти у заданому форматі.

Дана програма дозволяє побачити статистичні дані у наглядному вигляді, проаналізувати великий обсяг даних за короткий період часу, таким чином може робити рекомендації відносно конкретного клієнта магазина.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технологии и средства связи [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: http://www.tssonline.ru/articles2/fix-corp/rost-obema-informatsii--realii-tsifrovoy-vselennoy , вільн. - Див. з екрана.

2. Википедия [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://ru.wikipedia.org/wiki/Data_mining#Исторический_экскурс , вільн. - Див. з екрана.

3. Сайт Информационных технологий [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: http://inftech.webservis.ru/it/database/datamining/ar1.html , вільн. - Див. з екрана.

4. Интеллектуальные технологии поиска и анализа данных [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: http://www.study.urfu.ru/Aid/Publication/13334/1/Solonin.pdf , вільн. - Див. з екрана.

5. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining – СПб: БХВ-Петербург, 2004. – 336 с.

6. Сегаран Т. Программируем коллективный разум. – Пер. с англ. – СПб: Символ-Плюс, 2008. – 368 с.

7. Информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: http://www.machinelearning.ru , вільн. - Див. з екрана.

8. Чубукова И.А. Курс лекций «Data Mining», Интернет-университет информационных технологий [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: www.intuit.ru/department/database/datamining , вільн. - Див. з екрана.

9. Википедия Метод к-средних [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://ru.wikipedia.org/wiki/Метод_k-средних , вільн. - Див. з екрана.

10. Интеллектуальный анализ данных [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: http://intellect-tver.ru/?p=265 , вільн. - Див. з екрана.

11. Кластеризация: алгоритмы k-means и c-means [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://habrahabr.ru/post/67078/ , вільн. - Див. з екрана.

12. [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: http://cpp.com.ru/lippman/c20.html , вільн. - Див. з екрана.

13. Основные понятия интеллектуального анализа данных [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://docs.microsoft.com/ru-ru/sql/analysis-services/data-mining/data-mining-concepts , вільн. - Див. з екрана.

14. Методы интеллектуального анализа данных – IBM [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://www.ibm.com › Изучайте › Information Management , вільн. - Див. з екрана.

15. НОУ ИНТУИТ Лекция Интеллектуальный анализ данных [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://www.intuit.ru/studies/courses/2312/612/lecture/13260, вільн. - Див. з екрана.

16. Интеллектуальный анализ данных [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: elibrary.sgu.ru/uch_lit/1141.pdf, вільн. - Див. з екрана.

17. Data Mining [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://habrahabr.ru/hub/data_mining/ , вільн. - Див. з екрана.

18. Data Mining — добыча данных [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://basegroup.ru/community/articles/data-mining, вільн. - Див. з екрана.

19. Портал Знаний StatSoft [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: statistica.ru/local-portals/data-mining/, вільн. - Див. з екрана.

20. Data Mining, Лекция №1 – YouTube [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://www.youtube.com/watch?v=vtA93yys_20, вільн. - Див. з екрана.

21. Data Mining – Course [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://onlinecourses.nptel.ac.in/noc18_cs14, вільн. - Див. з екрана.

22. Обзор методов Data Mining Интеллектуальный анализ даннях [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: intellect-tver.ru/?p=165, вільн. - Див. з екрана.

23. Інтернаука «ИССЛЕДОВАНИЕ И АНАЛИЗ АЛГОРИТМА КЛАСТЕРИЗАЦИИ ДАННЫХ МЕТОДОМ К-СРЕДНИХ» [Електронний ресурс] // Комп'ютер-Інформ. – режим доступу: https://www.inter-nauka.com/issues/2017/18/3230 , вільн. - Див. з екрана.

# ДОДАТКИ

```
#pragma once
#include <math.h>
#include <vector>
#include <iostream>
#include <fstream>

namespace Диплом2018 {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Collections::Generic;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::Drawing::Drawing2D;
	using namespace System::IO;
	using namespace System::Windows::Forms::DataVisualization::Charting;
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
		}
	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
```

```
        ~MyForm()
        {
            if (components)
            {
                delete components;
            }
        }
    private: System::Windows::Forms::Button^  button1;
    protected:
    private: System::Windows::Forms::ProgressBar^  progressBar1;
    private: System::Windows::Forms::TextBox^  textBox1;
    private: System::Windows::Forms::MenuStrip^  menuStrip1;
    private: System::Windows::Forms::ToolStripMenuItem^
failToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^
saveAsToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^
exitToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^
helpToolStripMenuItem;
    private: System::Windows::Forms::ToolStripMenuItem^
aboutProgrammToolStripMenuItem;
    private: System::Windows::Forms::Button^  button2;
    private: System::Windows::Forms::TextBox^  textBox2;
    private: System::Windows::Forms::TextBox^  textBox3;
    private: System::Windows::Forms::Label^  label1;
    private: System::Windows::Forms::Label^  label2;
    private: System::Windows::Forms::DataVisualization::Charting::Chart^
chart1;
```

```
        private:
                /// <summary>
                /// Требуется переменная конструктора.
                /// </summary>
                System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
                /// <summary>
                /// Обязательный метод для поддержки конструктора - не
изменяйте
                /// содержимое данного метода при помощи редактора кода.
                /// </summary>
                void InitializeComponent(void)
                {

        System::Windows::Forms::DataVisualization::Charting::ChartArea^
chartArea2 = (gcnew
System::Windows::Forms::DataVisualization::Charting::ChartArea());

        System::Windows::Forms::DataVisualization::Charting::Legend^  legend2 =
(gcnew System::Windows::Forms::DataVisualization::Charting::Legend());
                        System::Windows::Forms::DataVisualization::Charting::Series^
series2 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Series());
                        this->button1 = (gcnew System::Windows::Forms::Button());
                        this->progressBar1 = (gcnew
System::Windows::Forms::ProgressBar());
```

```
            this->textBox1 = (gcnew
System::Windows::Forms::TextBox());
            this->menuStrip1 = (gcnew
System::Windows::Forms::MenuStrip());
            this->failToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->saveAsToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->exitToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->helpToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->aboutProgrammToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->button2 = (gcnew System::Windows::Forms::Button());
            this->textBox2 = (gcnew
System::Windows::Forms::TextBox());
            this->textBox3 = (gcnew
System::Windows::Forms::TextBox());
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->chart1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Chart());
            this->menuStrip1->SuspendLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->BeginInit();
            this->SuspendLayout();
            //
            // button1
```

```
                    //
                    this->button1->Location = System::Drawing::Point(12, 42);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(213, 23);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"File selection...";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // progressBar1
                    //
                    this->progressBar1->Location = System::Drawing::Point(12,
300);
                    this->progressBar1->Name = L"progressBar1";
                    this->progressBar1->Size = System::Drawing::Size(213, 23);
                    this->progressBar1->TabIndex = 1;
                    this->progressBar1->Value = 100;
                    //
                    // textBox1
                    //
                    this->textBox1->Location = System::Drawing::Point(12, 71);
                    this->textBox1->Name = L"textBox1";
                    this->textBox1->Size = System::Drawing::Size(213, 20);
                    this->textBox1->TabIndex = 2;
                    //
                    // menuStrip1
                    //
                    this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {
```

```
                this->failToolStripMenuItem,
                    this->helpToolStripMenuItem
            });
            this->menuStrip1->Location = System::Drawing::Point(0, 0);
            this->menuStrip1->Name = L"menuStrip1";
            this->menuStrip1->Size = System::Drawing::Size(738, 24);
            this->menuStrip1->TabIndex = 3;
            this->menuStrip1->Text = L"menuStrip1";
            //
            // failToolStripMenuItem
            //
            this->failToolStripMenuItem->DropDownItems-
>AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(2) {
                this->saveAsToolStripMenuItem,
                    this->exitToolStripMenuItem
            });
            this->failToolStripMenuItem->Name =
L"failToolStripMenuItem";
            this->failToolStripMenuItem->Size =
System::Drawing::Size(37, 20);
            this->failToolStripMenuItem->Text = L"File";
            //
            // saveAsToolStripMenuItem
            //
            this->saveAsToolStripMenuItem->Name =
L"saveAsToolStripMenuItem";
            this->saveAsToolStripMenuItem->Size =
System::Drawing::Size(152, 22);
            this->saveAsToolStripMenuItem->Text = L"Save as...";
            //
```

```
                    // exitToolStripMenuItem
                    //
                    this->exitToolStripMenuItem->Name =
L"exitToolStripMenuItem";
                    this->exitToolStripMenuItem->Size =
System::Drawing::Size(152, 22);
                    this->exitToolStripMenuItem->Text = L"Exit";
                    this->exitToolStripMenuItem->Click += gcnew
System::EventHandler(this, &MyForm::exitToolStripMenuItem_Click);
                    //
                    // helpToolStripMenuItem
                    //
                    this->helpToolStripMenuItem->DropDownItems-
>AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(1)
{ this->aboutProgrammToolStripMenuItem });
                    this->helpToolStripMenuItem->Name =
L"helpToolStripMenuItem";
                    this->helpToolStripMenuItem->Size =
System::Drawing::Size(42, 20);
                    this->helpToolStripMenuItem->Text = L"help";
                    //
                    // aboutProgrammToolStripMenuItem
                    //
                    this->aboutProgrammToolStripMenuItem->Name =
L"aboutProgrammToolStripMenuItem";
                    this->aboutProgrammToolStripMenuItem->Size =
System::Drawing::Size(156, 22);
                    this->aboutProgrammToolStripMenuItem->Text = L"About
program";
                    //
```

```
            // button2
            //
            this->button2->Location = System::Drawing::Point(12, 271);
            this->button2->Name = L"button2";
            this->button2->Size = System::Drawing::Size(213, 23);
            this->button2->TabIndex = 4;
            this->button2->Text = L"Processing";
            this->button2->UseVisualStyleBackColor = true;
            this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
            //
            // textBox2
            //
            this->textBox2->Location = System::Drawing::Point(12, 97);
            this->textBox2->Name = L"textBox2";
            this->textBox2->Size = System::Drawing::Size(37, 20);
            this->textBox2->TabIndex = 5;
            //
            // textBox3
            //
            this->textBox3->Location = System::Drawing::Point(12, 123);
            this->textBox3->Name = L"textBox3";
            this->textBox3->Size = System::Drawing::Size(37, 20);
            this->textBox3->TabIndex = 6;
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Location = System::Drawing::Point(55, 104);
            this->label1->Name = L"label1";
```

```
            this->label1->Size = System::Drawing::Size(20, 13);
            this->label1->TabIndex = 7;
            this->label1->Text = L"- K";
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->Location = System::Drawing::Point(55, 130);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(34, 13);
            this->label2->TabIndex = 8;
            this->label2->Text = L"- Limit";
            //
            // chart1
            //
            this->chart1->BorderlineColor =
System::Drawing::Color::Gray;
            this->chart1->BorderlineDashStyle =
System::Windows::Forms::DataVisualization::Charting::ChartDashStyle::Solid;
            chartArea2->AxisX->MajorGrid->LineColor =
System::Drawing::Color::Silver;
            chartArea2->AxisX2->LineColor =
System::Drawing::Color::Silver;
            chartArea2->AxisX2->MajorGrid->LineColor =
System::Drawing::Color::Silver;
            chartArea2->AxisY->MajorGrid->LineColor =
System::Drawing::Color::Silver;
            chartArea2->AxisY2->LineColor =
System::Drawing::Color::Silver;
```

```
chartArea2->AxisY2->MajorGrid->LineColor =
System::Drawing::Color::Silver;
chartArea2->BorderColor =
System::Drawing::Color::Gainsboro;
chartArea2->Name = L"ChartArea1";
chartArea2->ShadowColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<Syste
m::Byte>(224)), static_cast<System::Int32>(static_cast<System::Byte>(224)),

    static_cast<System::Int32>(static_cast<System::Byte>(224)));
this->chart1->ChartAreas->Add(chartArea2);
legend2->Name = L"Legend1";
this->chart1->Legends->Add(legend2);
this->chart1->Location = System::Drawing::Point(240, 42);
this->chart1->Name = L"chart1";
series2->BorderColor = System::Drawing::Color::Black;
series2->ChartArea = L"ChartArea1";
series2->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Point;
series2->EmptyPointStyle->Color =
System::Drawing::Color::Transparent;
series2->Font = (gcnew System::Drawing::Font(L"Microsoft
Sans Serif", 8));
series2->Legend = L"Legend1";
series2->MarkerSize = 7;
series2->Name = L"Series1";
this->chart1->Series->Add(series2);
this->chart1->Size = System::Drawing::Size(486, 281);
this->chart1->TabIndex = 9;
this->chart1->Text = L"chart1";
```

```
            //
            // MyForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(738, 335);
            this->Controls->Add(this->chart1);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->textBox3);
            this->Controls->Add(this->textBox2);
            this->Controls->Add(this->button2);
            this->Controls->Add(this->textBox1);
            this->Controls->Add(this->progressBar1);
            this->Controls->Add(this->button1);
            this->Controls->Add(this->menuStrip1);
            this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedSingle;
            this->MainMenuStrip = this->menuStrip1;
            this->Name = L"MyForm";
            this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterParent;
            this->Text = L"MyForm";
            this->menuStrip1->ResumeLayout(false);
            this->menuStrip1->PerformLayout();

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->EndInit();
            this->ResumeLayout(false);
```

```
            this->PerformLayout();

        }
#pragma endregion
    private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
                Stream^ myStream;
                OpenFileDialog^ openFileDialog1 = gcnew
OpenFileDialog;

                openFileDialog1->InitialDirectory = "C:\\";
                openFileDialog1->Filter = "txt files (*.txt)|*txt|All files
(*.*)|*.*";
                openFileDialog1->FilterIndex = 2;
                openFileDialog1->RestoreDirectory = true;

                if (openFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
                {
                    if ((myStream = openFileDialog1->OpenFile()) !=
nullptr)
                    {
                        //nameFileDialog = openFileDialog1-
>FileName;
                        myStream->Close();
                    }
                    textBox1->Text = openFileDialog1->FileName;

                }
    }
```

```
private: System::Void exitToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
                    Close();
}
            int num;
private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) {

                    if (textBox2->Text == "")
                    {
                        MessageBox::Show("Error! Fill in all the fields!",
"Error!", MessageBoxButtons::OK, MessageBoxIcon::Error);
                    }
                    else if (textBox1->Text == "")
                    {
                        MessageBox::Show("Error! Select the file!", "Error!",
MessageBoxButtons::OK, MessageBoxIcon::Error);
                    }
                    else
                    {

                        //отображение графика
                        num = Convert::ToInt32(textBox2->Text);
                        int i, j, k;
                        for (int u = 1; u <= num; u++)//удаление серии точек
                        {
                            chart1->Series->Remove();
                        }

                        for (j = 2; j <= num; j++)//добавление серии точек
```

```
                {
                    chart1->Series->Add("Series" + j); //заменить
имена на "Центроиды" и "имена колонок"
                    chart1->Series["Series" + j]->ChartType =
SeriesChartType::Point;
                    chart1->Series["Series" + j]->MarkerStyle =
MarkerStyle::Circle;
                    chart1->Series["Series" + j]->MarkerSize = 7;
                    chart1->Series["Series" + j]-
>MarkerBorderColor.Black;
                    chart1->Series["Series" + j]->MarkerBorderWidth
= 2;
                }

                for (j = 1; j <= num; j++)
                for (i = 0; i < 12; i++)
                {
                    chart1->Series["Series" + num]->Points-
>AddXY(mass1[i][0], mass1[i][1]); //AddXY(по X, по Y)
                }
            }
}
};
}

    #include <stdio.h>
    #include <stdlib.h>
    #include <math.h>

    typedef struct { double x, y; int group; } point_t, *point;
```

```
double randf(double m)
{
	return m * rand() / (RAND_MAX - 1.);
}

point gen_xy(int count, double radius)
{
	double ang, r;
	point p, pt = malloc(sizeof(point_t) * count);

	/* note: this is not a uniform 2-d distribution */
	for (p = pt + count; p-- > pt;) {
		ang = randf(2 * M_PI);
		r = randf(radius);
		p->x = r * cos(ang);
		p->y = r * sin(ang);
	}

	return pt;
}

inline double dist2(point a, point b)
{
	double x = a->x - b->x, y = a->y - b->y;
	return x*x + y*y;
}

inline int
nearest(point pt, point cent, int n_cluster, double *d2)
```

```
{
        int i, min_i;
        point c;
        double d, min_d;

#       define for_n for (c = cent, i = 0; i < n_cluster; i++, c++)
        for_n {
                min_d = HUGE_VAL;
                min_i = pt->group;
                for_n {
                        if (min_d > (d = dist2(c, pt))) {
                                min_d = d; min_i = i;
                        }
                }
        }
        if (d2) *d2 = min_d;
        return min_i;
}

void kpp(point pts, int len, point cent, int n_cent)
{
#       define for_len for (j = 0, p = pts; j < len; j++, p++)
        int i, j;
        int n_cluster;
        double sum, *d = malloc(sizeof(double) * len);

        point p, c;
        cent[0] = pts[ rand() % len ];
        for (n_cluster = 1; n_cluster < n_cent; n_cluster++) {
                sum = 0;
```

```
                for_len {
                        nearest(p, cent, n_cluster, d + j);
                        sum += d[j];
                }
                sum = randf(sum);
                for_len {
                        if ((sum -= d[j]) > 0) continue;
                        cent[n_cluster] = pts[j];
                        break;
                }
        }
        for_len p->group = nearest(p, cent, n_cluster, 0);
        free(d);
}

point lloyd(point pts, int len, int n_cluster)
{
        int i, j, min_i;
        int changed;

        point cent = malloc(sizeof(point_t) * n_cluster), p, c;

        /* assign init grouping randomly */
        //for_len p->group = j % n_cluster;

        /* or call k++ init */
        kpp(pts, len, cent, n_cluster);

        do {
                /* group element for centroids are used as counters */
```

```
                for_n { c->group = 0; c->x = c->y = 0; }
                for_len {
                        c = cent + p->group;
                        c->group++;
                        c->x += p->x; c->y += p->y;
                }
                for_n { c->x /= c->group; c->y /= c->group; }

                changed = 0;
                /* find closest centroid of each point */
                for_len {
                        min_i = nearest(p, cent, n_cluster, 0);
                        if (min_i != p->group) {
                                changed++;
                                p->group = min_i;
                        }
                }
        } while (changed > (len >> 10)); /* stop when 99.9% of points are
good */

        for_n { c->group = i; }

        return cent;
}

void print_eps(point pts, int len, point cent, int n_cluster)
{
#       define W 400
#       define H 400
        int i, j;
```

```
        point p, c;
        double min_x, max_x, min_y, max_y, scale, cx, cy;
        double *colors = malloc(sizeof(double) * n_cluster * 3);

        for_n {
                colors[3*i + 0] = (3 * (i + 1) % 11)/11.;
                colors[3*i + 1] = (7 * i % 11)/11.;
                colors[3*i + 2] = (9 * i % 11)/11.;
        }

        max_x = max_y = -(min_x = min_y = HUGE_VAL);
        for_len {
                if (max_x < p->x) max_x = p->x;
                if (min_x > p->x) min_x = p->x;
                if (max_y < p->y) max_y = p->y;
                if (min_y > p->y) min_y = p->y;
        }
        scale = W / (max_x - min_x);
        if (scale > H / (max_y - min_y)) scale = H / (max_y - min_y);
        cx = (max_x + min_x) / 2;
        cy = (max_y + min_y) / 2;

        printf("%%!PS-Adobe-3.0\n%%%%BoundingBox: -5 -5 %d %d\n",
W + 10, H + 10);
        printf( "/l {rlineto} def /m {rmoveto} def\n"
                "/c { .25 sub exch .25 sub exch .5 0 360 arc fill } def\n"
                "/s { moveto -2 0 m 2 2 l 2 -2 l -2 -2 l closepath "
                "       gsave 1 setgray fill grestore gsave 3 setlinewidth"
                " 1 setgray stroke grestore 0 setgray stroke }def\n"
        );
```

```
            for_n {
                printf("%g %g %g setrgbcolor\n",
                    colors[3*i], colors[3*i + 1], colors[3*i + 2]);
                for_len {
                    if (p->group != i) continue;
                    printf("%.3f %.3f c\n",
                        (p->x - cx) * scale + W / 2,
                        (p->y - cy) * scale + H / 2);
                }
                printf("\n0 setgray %g %g s\n",
                    (c->x - cx) * scale + W / 2,
                    (c->y - cy) * scale + H / 2);
            }
            printf("\n%%%%EOF");
            free(colors);
        #   undef for_n
        #   undef for_len
        }

#include "MyForm.h"
#include <iostream>
#include <conio.h>
#include <math.h>
#include <vector>


using namespace System;
using namespace System::Windows::Forms;
using namespace std;
        [STAThread]int main(array<String^>^ args)
```

```
        {
Application::EnableVisualStyles();
        Application::SetCompatibleTextRenderingDefault(false);
        Диплом2018::MyForm form;
                Application::Run(%form);
                int i;
                point v = gen_xy(PTS, 10);
                point c = lloyd(v, PTS, K);
                print_eps(v, PTS, c, K);
                // free(v); free(c);
                return 0;
}
```