Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

# Пояснювальна записка

до дипломної роботи

другий (магістерський)
(рівень вищої освіти)

на тему: Розробка мікросервісної аплікації з використанням нейромережі для надання рекомендацій

Виконав: студент 6 курсу групи КНз-61м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Лободинський Назар – Микола Ігорович
(прізвище та ініціали)

Керівник Левкович Мар'яна Володимирівна
(прізвище та ініціали)

Рецензент Демків Лідія Степанівна
(прізвище та ініціали)

Львів – 2022

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
_____ Крошний І. М.
"____"_____2022 року

# З А В Д А Н Н Я
## НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

*Лободинському Назарові-Миколі Ігоровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мікросервісної аплікації з використанням нейромережі для надання рекомендацій

керівник роботи *Левкович Мар'яна Володимирівна*,
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "4" березня 2021 року № #С-89

2. Термін подання студентом роботи 14 лютого 2022 року
3. Вихідні дані до роботи Постановка задачі та її формалізація

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
*1) провести попередній аналіз та обробку даних;*
*2) вибрати декілька моделей машинного навчання та оцінити якість роботи алгоритмів;*
*3)перевірити роботу кращого алгоритму на тестовій вибірці;*
*4)провести інтерпретацію отриманих результатів ;*
*5)розробити програмне забезпечення для представлення результатів роботи*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
*Знімки основних сторінок*

6. Дата видачі завдання 26 лютого 2021

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Огляд літературних джерел згідно із заданого завдання | 27.02.2021р-11.03.2021р | Виконано |
| 2 | Аналіз об'єкту дослідження та стану проблемної області:<br>1. Аналіз лосліджуваної проблеми.<br>2. Визначення вимог до системи | 12.03.2021р-01.04.2021р | Виконано |
| 3 | Вибірметодів і засобів для створення аплікації | 02.04.2021р-24.07.2021р | Виконано |
| 4 | Програмна реалізація системи:<br>1. Створення мікросервісної частини<br>2. Створення клієнтської частини | 25.07.2021р-20.11.2021р | Виконано |
| 5 | Відлагодження роботи програми, заповнення бази даних | 21.11.2021р-22.01.2022р | Виконано |
| 6 | Оформлення опису створеної програми | 23.12.2021р-30.12.2021р | Виконано |
| 7 | Здача пояснювалбної записки на перевірку та виправлення виявлених помилок | 17.01.2022р | Виконано |

**Студент** _____ **Лободинський Н-М. І.**
( підпис ) (прізвище та ініціали)

**Керівник роботи** _____ **Левкович М. В.**
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

В даній роботі проведено дослідження стосовно основних принципів будування мікросервіних систем та практично реалізовано за допомогою фреймворка .NET Core мікросервісну систему. Основною складовою проєтку є розробка рекомендаційного механізму за допомогою технологій машинного навчання. Виконаний механізм залучений в аплікацію як окремий сервіс. Рекомендаційний механізм отримано за допомогою використання бібліотеки ML.NET

# ABSTRACT

In this document, a study is conducted on the basic principles of building microservice systems and a microservice system is practically implemented using the .NET Core Framework . The main component of the project is the development of a recommendation mechanism using machine learning technologies. The executed mechanism is included in the application as a separate service. The recommendation mechanism is obtained by using the library ML.NET.

# ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити програмне та алгоритмічне забезпечення для рекомендаційного механізму з використанням машинного навчання, а саме:

1. провести попередній аналіз та обробку даних;
2. вибрати декілька моделей машинного навчання та оцінити якість роботи алгоритмів;
3. перевірити роботу кращого алгоритму на тестовій вибірці;
4. провести інтерпретацію отриманих результатів;
5. розробити сервісне забезпечення для представлення результатів роботи.
6. розробити клієнтське забезпечення для представлення результатів роботи.

# Зміст

# Перелік скорочень та умовних позначень

API – Application Programming Interface – Прикладний програмний інтерфейс

IT – інформаційні технології

SQL – мова структурованих записів

JSON – JavaScript Object Notation

DB – база даних

HTML – мова розмітки гіпертексту

DOM – Об’єктна модель документа

# Вступ

На сьогоднішній день розвиток технологій є приголомшливий, тому вже нікого не здивувати тривіальними технологіями. Сфера штучного інтелекту та машинного навчання все більше і більше збагачується дослідженнями, які започатковують нові проекти, що мають на меті спростити життя людям. У даній роботі було опрацьовано та досліджено одну з не складних задач машинного навчання, а саме ─ "надання рекомендацій". Рекомендації стали невід'ємною потребою під час проводження часу в інтернеті. Кожного разу, коли ми заходимо на YouTube, Netflix, Rozetka або онлайн магазини ми одразу помічаємо відібрані під нас відео чи товари.

Основною метою цієї дипломної роботи було застосування на практиці, відомих для багатьох людей трендів в розробці програмного забезпечення, а саме «мікросервісів» та «машинного навчання» для роботи над власним проєктом. Мікросервіси додають багато хороших властивостей для проєкту, такі як: простота заміни рішень/реалізацій, стійкість до відмов, зручність розробки (можна використовувати різні мови для розробки) та інше. Машинне навчання дозволяє нам додати виконання певної складної задачі в проєкт, в процесі виконання якої система «навчається» і це призводить до покращення результатів роботи програми. Результатом роботи став розроблений мікросервісний веб сайт для пошуку фільмів і окремий сервіс, який використовує алгоритм "колаборативної фільтрації" та надає рекомендації кіно для користувача.

*Актуальність* цієї теми зумовлена потребами бізнесу дати найкращі рекомендації для клієнтів. Клієнти в свою чергу отримують найбільш підходящі товари.

*Об'єкт дослідження* – засоби створення мікросрвісної системи, з окремим сервісом який уособлює рекомендаційний механізм.

*Предмет дослідження* – процес розроблення мікросрвісної системи з механізмом надання рекомендацій.

*Метою роботи* є розроблення розроблений мікросервісний веб сайт для пошуку фільмів і окремий сервіс, який використовує алгоритм “колаборативної фільтрації” та надає рекомендації кіно для користувача.

*Завдання*, які потрібно вирішити:

• реалізувати мікросервісну частину для системи;

• реалізувати сервіс для надання рекомендацій;

• реалізувати веб частину;

• протестувати роботу системи.

*Практичне значення* полягає у використанні веб-орієнтованої системи для пошуку фільмів та отримання рекомендацій, з можливістю використовування в комерційних цілях.

*Наукова новизна*. В процесі розроблення системи було використано сучасний архітектурний підхід для розробки, а саме мікросервісний. Механізм надання рекомендацій присутній в кожному популярному сервісі. Даний механізм є дуже важливий для бізнес частини в кожному комерційному роекті.

# РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

## 1.1 Мікросервісна архітектура

### 1.1.1 Визначення

Мікросервіси — стиль створення архітектури комп'ютерних проєктів, що реалізує шаблон сервісно-орієнтованої архітектури, який організовує систему як набір слабо пов'язаних між собою невеликих сервісів через легкі протоколи зв'язку (HTTP/HTTPS). Основна мета це - забезпечення незалежності окремих компонентів, які можуть бути розроблені незалежно від інших компонентів системи, а також чіткий поділ компонентів так, щоб реалізувати одну конкретну бізнес-логіку або логіка програмного забезпечення.

Архітектура мікросервісів чудово пасує для бізнес процесу безперервної доставки. Відмінністю сервіс-орієнтовної архітектури та мікросервісної є те що перша направлена для будування одного робочого застосунка, а сервісно орієнтована система — в свою чергу уособлює собою множину цілковитих застосунків які взаємодіють один з одним, тобто між собою.

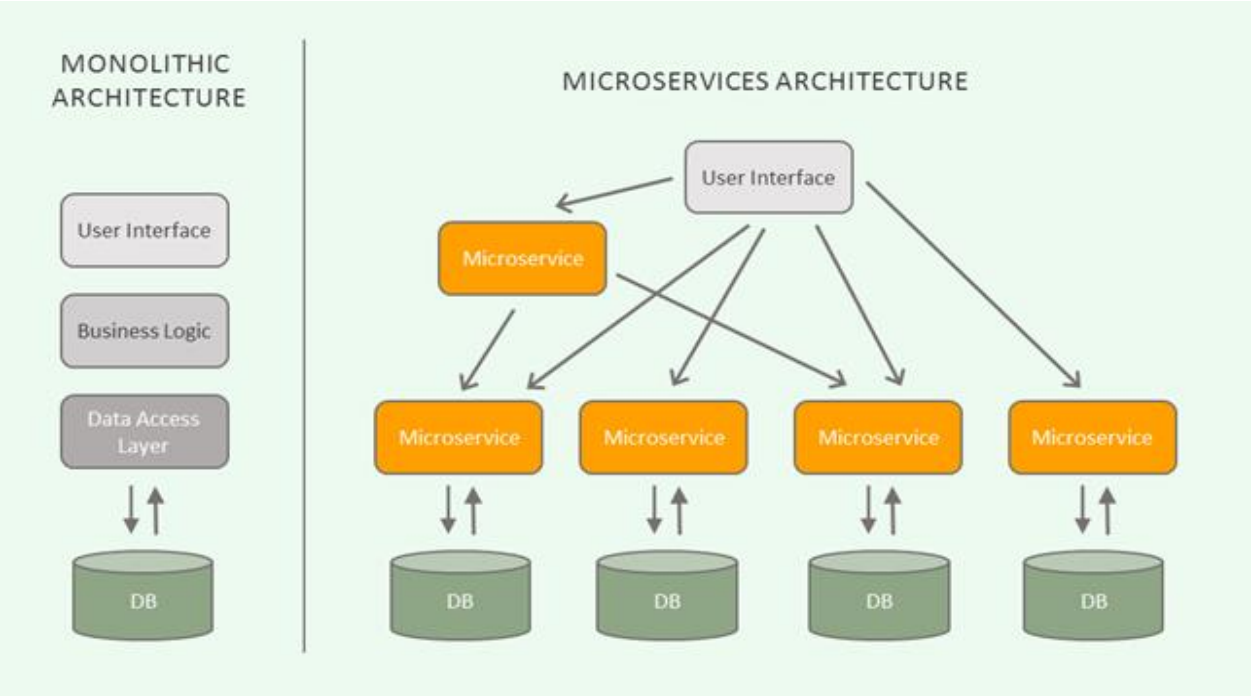


Рис.1. Графічне зображення мікросервісів

Точна дата створення концепції мікросервісів невідома. Будучи віце-президентом ThoughtWorks в 2004 році, Фред Джордж почав працювати над прототипом архітектури, заснованим на тому, що він назвав “принципами Baysean”, названими на честь відомого Джеффа Бея

### 1.1.2 Основні властивості та філософія

Властивості мікросервісного стилю архітектури:

- можна працювати над різними сервісами, таким чином розпаралелити роботу
- позволяє швидко залучувати нових програмістів та зменшує кількість потенційних конфліктів
- Можливість реалізування на одній та шншій мові програмування, СБД, та ін.

- Повинні мати опрацьовані потенційні вразливості системи
- Кожен сервіс гнучкий,  легко компонується та взаємодіє з іншими сервісами, стійкий до відмов та вразливостей,  функціонально мінімальний та завершений.
- Немає залежності від конкретної технології та одного постачальника
- Простота проектування і внесення змін в окремі елементи,
- Збій не впливає на всю систему
- добре для великих систем

### 1.1.3 Недоліки

В мікросервісній архітектурі критики в основному називають наступні проблеми:

- CAP теорема, фінальна узгодженість
- Доволі високі фінансові витрати на підтримку інфраструктуру, моніторинг/сервіс і операційні дії
- Важко досягти згоди між програмістами тямущо: різні точки зору архітектуру.
- Ускладнено зреалізувати тестування цілковитої системи і додатково розгортання
- Ускладнено зреалізувати забезпечення цілковитої та надійної безпеки

# 1.2 Клієнт-серверна архітектура

Розробка програмного забезпечення є одним з найскладніших параметрів в IT. Завдяки дизайну цих програм ми можемо сьогодні говорити і виконувати завдання, які раніше були неможливі, від перегляду Інтернету до обміну документом з іншим комп'ютером в тій же мережі. Що таке архітектура програмного забезпечення (Software Architecture)?

Згідно з популярним визначенням, архітектура програмного забезпечення-це спосіб організації системи, особливо це стосується її компонентів, робочого середовища і правил, що визначають спосіб її побудови і розвитку. Архітектура програмного забезпечення-це спосіб забезпечення надійності, гнучкості, безпеки, масштабованості системи, а також спосіб задоволення технічних і бізнес-очікувань, що пред'являються до неї. Архітектура програмного забезпечення також визначає відносини, взаємодії, що відбуваються між елементами системи, тому відноситься до її структури. Визначає завдання, які будуть виконуватися розробниками. Являє собою своєрідну схему побудови, при якій окремі компоненти точно визначаються, визначаються. Головною метою архітектури програмного забезпечення є створення системи, яка буде працювати, вести себе відповідно до різних очікувань. Таким чином, це також спосіб їх гармонізації, оркестровки.

Парадигма клієнт-сервер описує асиметричні відносини між двома процесами, що протікають зазвичай в різних місцях:

процес сервер:

- управляє деякими ресурсами в місці а;
- пропонує послуги, доступні для інших процесів;
- при запуску працює протягом тривалого періоду часу;
- пасивно очікує на прохання клієнта;

- виконує завдання, відповідаючи на повідомлення  клієнт

процес клієнт:

- він потребує конкретної Послуги для виконання свого робота на місці Б;
- відправляє запит на сервер для виконання конкретного завдання в сфері його послуг;
- отримує з сервера результат цього завдання

Приклад: Сервер управляє базою даних; Пропонована послуга: доступ до бази даних; Клієнт отримує доступ до якогось певного запису в базі даних

У парадигмі клієнт-сервер паралелізм відсутній, це скоріше метод структурування програм і метод інтеграції для різноманіття ресурсів.

Приклад: У великих додатках поділ між клієнтом і сервером часто є розділом між різними програмами розроблені різними авторами і навіть різними компаніями. Розглянута парадигма однозначно визначає інтерфейс між цими програмами.

Однак паралельність не виключена:

- на стороні клієнта може застосовуватися багатопоточність, T. J. система може перейти до виконання іншого завдання, у той час як сам клієнт очікує відповіді сервера;
- також сервер може виконувати інші завдання в очікуванні повідомлення клієнта.

Клієнт-серверні системи зазвичай будуються на основі архітектур LAN / WAN, що утворюють слабо пов'язані розподілені системи. Отже, в цих системах застосовується більше, ніж в системах паралельна вага для безпеки, толерантності до помилок, гетерогенності і сумісності .Клієнт-серверні системи на відміну від паралельних систем відрізняється динамічністю. Розробник не виходить з одного чітко визначеного завдання для вирішення і специфікації програми, що вам потрібно розділити на обчислювальні процеси. Програмне

забезпечення сервера знає інтерфейс, але не знає кількість клієнтів, або сукупності всіх програм, які в майбутньому будуть використовувати служби сервера. Клієнти також не знають один одного.

Таким чином, клієнт-серверні системи більш непередбачувані. Клієнт-серверні системи зазвичай працюють у моделі MPMD, хоча багато клієнтів можуть виконувати одну і ту ж програму. Клієнтські та серверні програми використовують окремі простори адресів та спілкуються через обмін повідомленнями. Конкретні механізми відрізняються від простих send / receive .Типовими ресурсами серверів є файлові системи, бази даних, складні програми, принтери і т. д. Сервер знаходиться зазвичай поруч з ресурсами. Більшість серверів співпрацюють з кількома клієнтами. Один клієнт може співпрацювати протягом свого періоду роботи з кількома серверами. Сервер-виконуючи доручене завдання-може користуватися послугами інших серверів. Сервер з декількома клієнтами може працювати ітеративним чином.

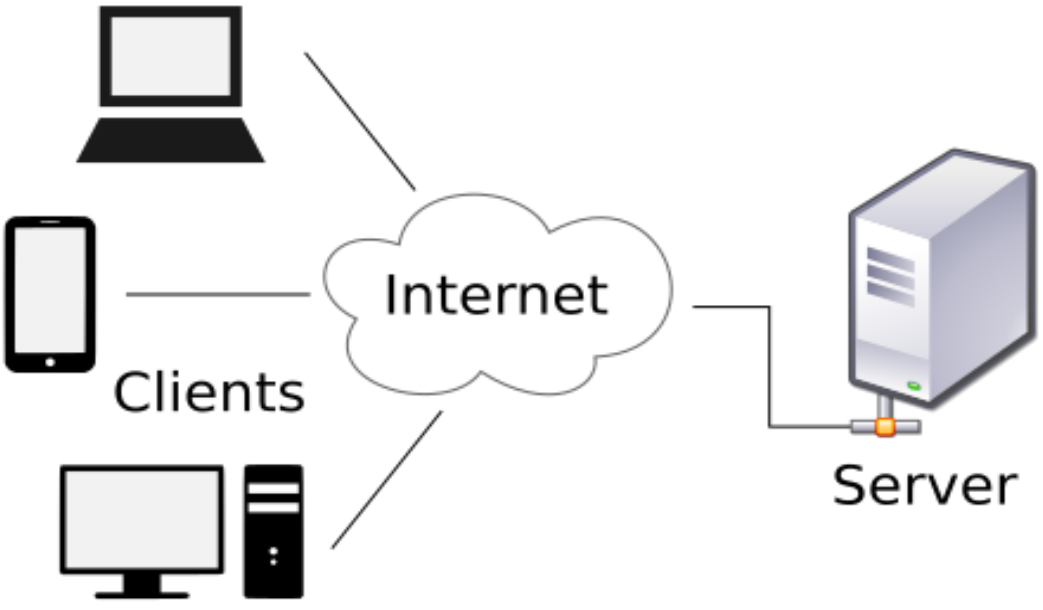


Рис.2. Модель клієнт-серверної архітектури

## 1.3 .NET Framework

Microsoft .NET (читати дот-нет) — платформа розробки, розроблена компанією Microsoft, включає в себе середовище виконання (CLR) і бібліотеки класів, що надають стандартні функціональні можливості для додатків. Ця технологія не пов'язаний з конкретною мовою програмування, а програми можуть бути написані на одному з багатьох мов, наприклад, C++/CLI, C#, F#, J#, Delphi 8 для .NET, Visual Basic .NET. Завдання платформи .NET Framework є управління різними елементами системи: код програми, пам'яттю і безпекою.

Що це таке framework - З терміном "framework" ми можемо зустрічатися кілька разів, навіть якщо ми не займаємося щоденним програмуванням. Все завдяки наявності. NET Framework, який є компонентом операційних систем Microsoft Windows. Але спочатку варто з'ясувати, що таке фреймворк? Ми визначаємо його як платформу розробки, яка є основою для створення Програми, в першу чергу містить структуру створюваного додатка і його загальну схему роботи. Framework також містить набір бібліотек і компонентів, які можуть бути застосовані для виконання різних завдань. При програмуванні доступні компоненти повинні бути розширені і адаптовані до потреб підготовлюваного проекту, щоб в результаті було запущене додаток.

Чи є. NET Framework необхідним? Так, так. Якщо ми повністю не відмовимося від програмного забезпечення, що вимагає наявності .NET. більшість новітніх продуктів Microsoft, а також сторонніх виробників, створюються на основі. Net, щоб не виникало проблем із сумісністю, Windows Vista має основні компоненти, встановлені за замовчуванням. Звичайно, з розвитком мов програмування, апаратних і системних можливостей, з'являються все нові і нові версії бібліотек. NET Framework, установка яких іноді необхідна. У попередньому розділі статті ми пояснювали, що таке. NET Framework, в той час як в цьому розділі ми виклали необхідність мати його в комп'ютері . Для більшості користувачів-власникам Windows, знання про можливості. Net не потрібні, тому що більшість програм,

які вимагають присутності платформи, автоматично її не встановлять або раніше завантажили з Інтернету. Однак для розробників, які використовують Visual Studio, бібліотеки. NET Framework є безцінним зручністю. Уявіть, що ми-людина, яка пише програму, яка вимагає втручання у функції системи. Програмне забезпечення для такої події вимагає написання десятків рядків коду, і за допомогою. Net ми можемо виконати це завдання, ввівши кілька рядків інструкцій. Для компілятора це не має великого значення, чи використовує він" готові", реалізовані на платформі, або переводить інструкції розробника. Однак для користувача такої програми або розробника, що створює додаток, це має першорядне значення. З. NET Framework реалізація багатьох функцій стає простіше, швидше і, можливо, більш оптимальною. З іншого боку, одержувач отримує програму, яка характеризується більшою стабільністю, ефективністю і меншим споживанням апаратних ресурсів.

У серпні 2000 року Microsoft, Hewlett-Packard і Intel спільно представили специфікації common Language Infrastructure і C # в ECMA в якості стандартної пропозиції. Робота над ними проводилася в рамках Комітету TC39 в підгрупах TG3 і TG2, за участю m.in. IBM і Fujitsu. Вони були остаточно затверджені в грудні 2001 року як ECMA - 334 (C#) і ECMA-335 (CLI), а технічний опис як TR/84, а потім передані для прийняття ISO. У квітні 2003 року ISO визнала відправлені стандарти, надавши їм номери ISO / IEC 23270 (C#), ISO/IEC 23271 (CLI) та ISO/IEC 23272 (CLI TR), а ECMA прийняла їх в якості другого випуску своїх стандартів.

## 1.4 C#

Що таке C#? Це високорівнева об'єктно-орієнтована мова програмування, створена Microsoft з дуже широким спектром застосування. Об'єктно-орієнтоване програмування якимось чином змодельовано на роботі людського мозку, дозволяє пов'язати дані з певною процедурою, покликане полегшити

написання, обслуговування і багаторазове використання програм або їх фрагментів. Мова C# пов'язана з платформою. Net, яка є платформою та середовищем виконання. Колись єдиним основним призначенням мови C# було написання додатків, які повинні були підтримувати Windows, сьогодні, завдяки доступності платформи. NET на таких системах, як Linux або Mac, можна створювати їх на всіх платформах, що робить його дуже універсальною мовою . Крім того, він вважається досить легко засвоюваним.

Для чого використовується c#? Завдяки тому, що мова C# є постійно розвивається мовою Microsoft, він залишається на передньому краї найпопулярніших мов, використовуваних для програмування. Його застосування досить широке - від створення ігор типу PokemonGO або Angry Birds, до великих, великих систем, використовуваних банками або логістичними компаніями, до програмування ботів і безпілотних літальних апаратів.

Назва мови виникла за аналогією з ім'ям C++. Оператор ++ в C (в C++ теж) означає збільшення на одиницю, тому C++ більше, ніж C. автори C# використовували аналогічну ідею, де символ перехрестя / решітки нагадує чотири взаємопов'язаних оператора +. У музиці звук C♯ вище, ніж звук C, це може означати, що мова є розвитком C / C++.Символ ♯ (хрест) недоступний на більшості клавіатур, деяких шрифтів і веб-браузерів, тому рекомендується використовувати символ решітки ( # ), який дуже схожий на музичний символ. Іноді, коли це виправдано, наприклад, в рекламних матеріалах, Microsoft використовує Ім'я C♯ замість C#.

# 1.5 ASP.net

Інтернет-мережа багатьма вважається одним з найважливіших винаходів другої половини минулого століття. Вона використовується в різних цілях значною частиною населення. Веб-сайти (World Wide Web) є одним з найпопулярніших і часто використовуваних сервісів, присутніх в Інтернеті. За свою довгу кар'єру вони зазнали своєрідну революцію. Спочатку вони містили тільки статичний текст, а пізніше, завдяки таким технологіям, як html, на веб-сайтах стала з'являтися графіка, також був введений революційний у своїй простоті механізм Kiper-посилань. Основним недоліком тодішніх веб-сайтів була відсутність будь-якої можливості впливати на користувача браузера на їх зміст.

ASP.NET -технологія розробки веб-додатків і веб-сервісів Microsoft. Є складовою частиною платформи Microsoft.NET і розвитком застарілої технології Microsoft ASP. В даний час остання версія цієї технології ASP.NET Core 3.0

ASP.NET зовні багато в чому зберігає схожість з більш старою технологією ASP, що дозволяє розробникам відносно легко перейти на ASP.NET. у той же час внутрішній устрій ASP.NET принципово відрізняється від ASP, оскільки вона заснована на платформі. NET і тому використовує всі нові можливості, пропоновані цією платформою.

Хоч ASP.NET він бере свою назву від старої технології ASP від Microsoft, сильно відрізняється від неї. Розробники можуть писати код для ASP.NET за допомогою майже всі мов програмування, що входять в комплект поставки. NET Framework (C#, Visual Basic.NET і JScript.NET). ASP.NET кращий порівняно з технологіями сценаріїв, оскільки при першому доступі код компілюється і поміщається в спеціальний кеш, а потім виконується тільки без необхідності витрачати час на розбирання, проведення оптимізації і т. д. Переваги ASP.NET:

- ASP.NET він має перевагу швидкості в порівнянні з іншими технологіями на основі сценаріїв (PHP і т. д.)
- Розширюваний набір елементів управління і бібліотек класів дозволяє швидше створювати додатки
- ASP.NET він заснований на багатомовних можливостях Netflix, YouTube, що дозволяє писати код сторінки в C#, VB, C / C++ і т. д.
- Фактичний поділ цілої бізнес-логіки та візуальної частини

## 1.6 NET Core

ASP.NET Core — це безкоштовний веб-фреймворком з відкритим вихідним кодом і наступником ASP.NET , розроблений корпорацією Майкрософт. Це модульна структура, яка працює як на повній, так і на полной.NET Framework, в Windows, і крос-платформний. NET. однак ASP.NET базова версія 3 працює тільки на.NET Core, відмовляючись від підтримки Платформи. NET Framework.

Фреймворк являє собою нову повністю переписану версію, що об'єднує раніше окремі ASP.NET MVC і ASP.NET Веб-API в єдину модель програмування.

Незважаючи на те, що це новий фреймворк, побудований на новому веб-стеку, він має високий ступінь сумісності концепції з ASP.NET . В ASP.NET базова платформа підтримує паралельне управління версіями, так що різні додатки, що розробляються на одній машині, можуть бути націлені на різні версії ASP.NET Ядро. Це неможливо в попередніх версіях ASP.NET .

Blazor-це недавній (додатковий) компонент для підтримки WebAssembly, і починаючи з версії 5.0 він припиняє підтримку деяких старих веб-браузерів. У

той час як поточний Microsoft Edge працює, його застаріла версія, тобто "Microsoft Edge Legacy" і Internet Explorer 11, видаляються при використанні Blazor.

## 1.7. ML.NET

ML.NET - це дармова бібліотека машинного навчання програмного забезпечення для мов програмування C # і F # . Ця бібліотека підтримує моделі Python , якщо вони використовуються в парі з NimbusML. Попередній випуск ML.NET включав перетворення для функціональної інженерії, наприклад створення n-грамів , і учнів, які мають справу з регресійними завданнями , двійковою класифікацією та багато класовою класифікацією. Від того часу було додано додаткові завдання в галузі ML, такі як системи виявлення аномалій та рекомендації, а інші підходи, такі як глибоке навчання, будуть включені в наступні версії.

## 1.8 Хмарна платформа Heroku

Heroku є однією з перших хмарних платформ, які працюють як Platform as a Service (PaaS). Його робота заснована на контейнерах, що використовуються для розміщення веб-програми в serverless. Спочатку він був заснований на мові програмування Ruby, однак сьогодні він також підтримує вузол. js, Java, PHP, Python, йти, Scala або Clojure. Він забезпечує швидку і відносно просту установку і управління, а також автоматичне масштабування сучасних додатків без необхідності мати справу з інфраструктурою і конфігурацією серверів. Під час компіляції Мовні і системні стеки постійно оновлюються і контролюються, що запобігає Конфліктам і гарантує, що програма готова до роботи.

Heroku — хмарна PaaS-платформа, яка підтримує багато мов програмування. Компанією Heroku володіє Salesforce.com. Дана компанія одна з перших хмарних платформ, з'явилась на ринку в червні 2007 року. 15 вересня 2011 року Facebook та Heroku представили новий продукт «Heroku для Facebook». Heroku також має в арсеналі підтримку таких систем для управління базами даних, як CouchDB, Membase, MongoDB і Redis, крім основної — PostgreSQL.

Проєкти, які працюють на Heroku, використовують DNS-сервер Heroku (переважно додатки мають доменне ім'я по типу «ім'я_додатку.Herokuapp.com»). Під кожну програму виділяється кілька процесів, які мають назву «dynos».

Переваги використання Heroku:

- дружній CLI, за допомогою якого ви можете завантажити додатки, перевірити журнали і стан програми
- Безкоштовно Ви отримуєте 500 годин роботи dyno, який є мікро інстанцією, на якій працює додаток. Ліміт можна подвоїти вдвічі, якщо ввести платіжну картку на рахунок. Тайм-аут відноситься до фактичного часу використання серверів у вашому обліковому записі, тому він враховується тільки в тому випадку, якщо хтось дійсно використовує ваш додаток.
- Ви можете підключити свій домен безкоштовно
- За Heroku стоїть Salesforce-гігант коли справа доходить до створення програмного забезпечення, тому вам не потрібно турбуватися про те, що служба раптово перестане працювати і ви втратите всі файли. Крім того, Heroku стоїть на AWS, найбільшому хостинг-провайдері в світі.

Мета Heroku також полягає в тому, щоб полегшити роботу розробників і допомогти вирішити найбільш поширені проблеми. Він поставляється з інструментами для автоматичного повідомлення про помилки і масштабування додатків, якщо вони перевищують тимчасові пороги, встановлені користувачем.

# 1.9 MS SQL Server

Перш ніж ми перейдемо до обговорення основ адміністрування Microsoft SQL Server, варто зупинитися і подумати про те, що це за продукт насправді. SQL Server часто розглядається користувачами як база даних. Хоча це визначення частково вірно, воно, тим не менш, не повністю відображає повний характер цієї системи. Так, реалізований в ньому движок бази даних, безсумнівно, є його найважливішою частиною, проте він є лише одним з багатьох основних компонентів, які разом складають так звану систему управління реляційною базою даних (англ. Relational Database Management System, СУБД).

Реляційна база даних-це база даних, в якій дані збираються і зберігаються в двовимірних об'єктах, близьких за структурою до таблиць. Згідно з конкретним проектом бази даних, на ці таблиці накладаються певні обмеження (constraints), що забезпечують цілісність даних, а між обраними таблицями визначаються з'єднання, які надають цим даним відповідний сенс. В основі роботи реляційної бази даних лежить математична модель організації даних, заснована безпосередньо на теорії множин і понятті відносин. У цій моделі дані групуються в таблиці, звані відносинами, між якими створюються вже згадані відносини, які надають цим даним правильний контекст і сенс. Про реляційної моделі даних і використовуваної в ній термінології ми розповімо докладніше при обговоренні теми проектування баз даних.

Таким чином, SQL Server-це система СУБД, реалізована і працює в архітектурі клієнт / сервер, де кожен компонент здатний працювати самостійно і незалежно від інших. Завдяки взаємної співпраці ці компоненти реалізуються, як правило, у вигляді фонових послуг, дозволяють нам не тільки збір, обробку, збереження, аналіз і звітність даних з використанням високопродуктивних і оптимізованих операцій, але і, відповідно, захищають їх від несанкціонованого доступу з використанням поширених механізмів аутентифікації, спрощують резервне

копіювання і відновлення даних, а також дозволяють автоматизувати операції і завдань відповідно до певного графіка.

Підтримка системи SQL Server здійснюється в основному за допомогою спеціальних інструментів управління, оснащених графічним інтерфейсом, при цьому управляти роботою сервера і движка бази даних можна також за допомогою відповідних команд командного рядка і оболонки Windows PowerShell. Однак основним засобом зв'язку Користувача з движком бази даних є структурована мова запитів (Structured Query Language, SQL) декларативно-імперативного характеру, який через наявність безлічі власних розширень Microsoft присутній тут на діалекті Transact-SQL (T-SQL). За допомогою T-SQL ми можемо не тільки реалізовувати і взаємодіяти з обраною базою даних, але і управляти роботою самого сервера бази даних.

## 1.10 Docker

Docker в даний час є важливим інструментом для більшості розробників. Згідно з дослідженням Stack Overflow, Docker-це технологія, знання якої найбільш затребуване розробниками. Отже Docker -це набір інструментів для управління ізольованими контейнерами Linux. Docker в цілому доповнює інструментарій lxc більш високим рівнем Ari, дозволяючи управляти контейнерами на рівні ізоляції окремих незалежних процесів. Зокрема, Docker дозволяє, повністю не вникаючи в вміст контейнера, запускати довільні процеси в режимі ізоляції, а потім переносити і клонувати контейнери, створені для цих процесів, на інші сервери, після завершення всіх робіт, пов'язаних зі створенням, обслуговуванням і збереженням контейнерів.

Вихідний код проєкту docker написаний на мові го і поширюється під ліцензією Apache 2.0. Набір інструментів повністю заснований на використанні

стандартних механізмів ізоляції, котрі вбудовані в ядро Linux, на основі просторів імен (Імен) і контрольних груп (cdoirs). Сценарії Lxc використовуються для створення контейнерів. Щоб створити контейнер, просто потрібно завантажити образ нашого базового середовища (команда Docker pull base), після чого ви зможете запускати, будь-які, програми в пісочниці (наприклад, для запуску bash ви можете запустити Docker run-i-T base / bin / bash).

Основні можливості Docker:

- Можливість розміщення в ізольованому середовищі різного заповнення, включаючи різні комбінації виконуваних файлів, бібліотек, конфігураційних файлів, сценаріїв, файлів jar, gem, tar і т. д
- Підтримка роботи на будь-якому комп'ютері з архітектурою x86_64 на базі ядра Linux, від ноутбуків до серверів і віртуальних машин. Можливість роботи з модифікованими сучасними ядрами Linux (без накладення патчів) і в стандартних середовищах всіх основних дистрибутивів Linux, включаючи Fedora, RHEL, Ubuntu, Debian, SUSE, Gentoo і Arch;
- Оскільки контейнери використовують власну автономну файлову систему, не має значення, де, коли і в якому середовищі вони запускаються.
- Використання легких контейнерів для ізоляції процесів від інших процесів та основної системи.
- Ізоляція на рівні мережі: кожен ізольований процес має доступ тільки до простору імен мережі, пов'язаного з контейнером, включаючи віртуальний мережевий інтерфейс і пов'язані IP-адреси;
- Ізоляція на рівні файлової системи: кожен процес виконується в повністю окремому кореневому FS;

- Ізоляція ресурсів : використання системних ресурсів, таких як втрата пам'яті та завантаження процесора, може бути обмежено окремо для кожного контейнера за допомогою cgroups;

- Основна файлова система для контейнерів створюється за допомогою copy-on-write [en] (окремо зберігаються тільки змінені і нові дані), що прискорює розгортання, скорочує використання пам'яті і економить дисковий простір;

- Підтримка використання різних систем зберігання, які можуть бути підключені як вилки. Серед підтримуваних драйверів пам'яті оголошені agaihut, device mapper [en] (використовувані знімки LVM), vs (на основі копіювання каталогів) і Btrfs. Очікується, що драйвери для ZFS, Gluster і Ceph;

- Всі стандартні потоки (stdout / stderr) кожного процесу, що виконується в контейнері, збираються і зберігаються як журнал;

- Модифікована файлова система одного контейнера може використовуватися в якості основи для створення нових базових зображень і створення інших контейнерів без необхідності проектування шаблонів або ручного налаштування теми зображень;

- Можливість створення контейнерів, що містять складні стеки програмного забезпечення, шляхом об'єднання вже існуючих контейнерів, що містять елементи створюваного стека. Об'єднання відбувається не шляхом об'єднання вмісту, а шляхом забезпечення сумісності між контейнерами (створюється мережевий тунель).

- Можливість використання інтерактивної командної оболонки: до стандартного входу будь-якого контейнера можна прив'язати псевдо-tty для запуску оболонки.

Docker складається з двох процесів:

- Зображення Docker-містить операційну систему, додаток і всі його залежності. Зображення в Docker складаються з шарів. Якщо нам потрібне зображення з веб-

сервером, Ми беремо за основу зображення з дистрибутив операційної системи, додаємо залежність-веб-сервер і зберігаємо це як нове зображення, яке буде мати два шари-один з операційною системою, інший з веб-сервером. Зображення можуть бути розділені через DockerHub.

- Демон Docker, який працює на гостьовому комп'ютері (якщо це Linux) або всередині VirtualBox boot2docker (якщо це Windows або OS X).

- Клієнт, за допомогою котрого можна взаємодіяти з демоном .

- Контейнер Docker-це робоче зображення . Контейнер Docker можна запускати, зупиняти, переміщати і видаляти. Ви також можете зробити Docker commit контейнер, який створить образ з поточного стану контейнера.

  Навіщо нам потрібно використовувати Докер - Оскільки контейнери Docker містять все, що потрібно для запуску програми (і тільки ці речі), вони дозволяють легко переміщати програми між середовищами. Будь-який, хост із встановленим середовищем виконання Docker - будь то, ноутбук розробника, програміста або екземпляр загальнодоступної хмари-може запустити контейнер Docker.

  Кілька слів підсумування: Ви можете чітко бачити, що середовище розробки може бути побудоване з готових зображень, що містять встановлені контейнери та служби. Ми також можемо використовувати механізм створення контейнерних образів і надавати наші конкретні програми у вигляді зображень. Незалежно від технології, використовуваної в додатку, зображення контейнера буде поставлятися ідентичним чином, що спрощує процедури установки.

# РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

## 2.1 Машинне навчання

Машинне навчання-це галузь штучного інтелекту та інформатики, яка фокусується на використанні даних і алгоритмів для імітації того, як люди вчаться, поступово підвищуючи точність.

IBM одна з перших компаній яка має багатий досвід роботи з машинним навчанням. Артур Самуель визнаний автором терміна "Машинне навчання", який він використовував в одному зі своїх досліджень про шашки. Роберт Ніл, самопроголошений чемпіон з шашок, програв комп'ютеру IBM 7094 в 1962 році. У порівнянні з сьогоднішніми можливостями цей подвиг здається тривіальним, однак він вважається важливою віхою в області штучного інтелекту. Протягом наступних десятиліть технологічні розробки в області зберігання та обчислювальної потужності дозволили створити інноваційні продукти, які ми знаємо і любимо, такі як Netflix або автономні автомобілі.

Машинне навчання є важливою частиною зростаючої галузі аналітики великих даних. Використовуючи статистичні методи, алгоритми навчаються класифікувати або прогнозувати при наданні ключових ідей в рамках проектів інтелектуального аналізу даних. Ці ідеї потім використовуються при прийнятті рішень про додатки та бізнес, що відмінно впливає на ключові показники зростання. У міру того, як великі набори даних продовжують розширюватися і зростати, ринковий попит на аналітиків даних буде рости, а також необхідність допомогти визначити найбільш важливі бізнес-питання, а потім дані, які дадуть відповідь на них.

Оскільки терміни "глибоке навчання" і "машинне навчання" зазвичай використовуються взаємозамінно, варто звернути увагу на незначні відмінності,

що розділяють їх. Машинне навчання, глибоке навчання та нейронні мережі є підрозділами штучного інтелекту. Більш конкретно-глибоке навчання, однак, являє собою субпідряд машинного навчання, а нейронні мережі-субпідряд глибокого навчання.

Різниця між глибоким і машинним навчанням полягає в тому, як кожен алгоритм навчається. Глибоке навчання автоматизує значний обсяг процесу вилучення ознак, усуваючи частину необхідного ручного втручання людини і дозволяючи використовувати більші набори даних. Глибоке навчання можна розглядати як "масштабоване Машинне навчання", як зазначив Лекс Фрідман в лекції MIT. Класичне, тобто "нелогічне" Машинне навчання більшою мірою залежить від втручання людини. Експерти визначають набір характеристик, щоб зрозуміти відмінності між вхідними даними. Зазвичай для навчання потрібні більш структуровані дані.

"Глибоке" Машинне навчання може використовувати набори даних з мітками в рамках так званого контрольованого навчання для інформування алгоритму, але не вимагає набору даних з мітками. Він може ґрунтуватися на неструктурованих даних в необробленій формі (наприклад, тексті, зображеннях) і автоматично визначати набір характеристик, які розрізняють категорії даних. На відміну від машинного навчання для обробки даних, воно не вимагає втручання людини, що дозволяє масштабувати Машинне навчання більш цікавими способами. Глибоке навчання і нейронні мережі в першу чергу визнані прискорюють прогрес в таких областях, як розпізнавання зображень, обробка природної мови і розпізнавання мови.

Нейронні мережі або штучні нейронні мережі складаються з шарів вузлів, що охоплюють вхідний шар, один або кілька прихованих шарів і вихідний шар. Кожен вузол (штучний нейрон) з'єднується з іншим і має пов'язаний вага і поріг. Якщо вихід будь-якого окремого вузла перевищує задане порогове значення, цей вузол активується при відправці даних на наступний рівень мережі. В іншому

випадку дані не передаються на наступний рівень мережі. Слово "глибокий" в контексті глибокого навчання відноситься саме до рівнів шарів в нейронній мережі. Нейронна мережа, що складається з більш ніж трьох шарів, включаючи вхідний і вихідний шари, може розглядатися як алгоритм глибокого навчання або глибока нейронна мережа. Нейронна мережа, яка має лише два або три шари, - це лише основна нейронна мережа.

Кращі мови програмування для машинного навчання: Мови програмування використовуються для написання інструкцій з навчання систем машинного навчання. Кожен з них об'єднує співтовариство користувачів, від яких можна багато чому навчитися. І кожен містить спеціальні бібліотеки для машинного навчання. Отто 10 найпопулярніших з них за даними порталу GitHub. 10 найпопулярніших мов за опитуванням 2019 року. - Python C++ JavaScript Java C # Julia Shell R TypeScript Scala-використовується для роботи з великими наборами даних

## 2.2 Задачі машинного навчання

У сьогоднішньому бізнесі Машинне навчання має дуже широкий спектр застосувань, які з часом, безумовно, будуть розширюватися. Підрозділи ML включають m.in. соціальні мережі та рекомендації по продуктам, розпізнавання зображень, медична діагностика, переклад мови, розпізнавання мови і Data mining.

Платформи соціальних мереж, такі як Facebook, Instagram та LinkedIn, використовують ML, щоб запропонувати сторінки, на які ви можете слідкувати, на основі вподобаних вами повідомлень, або групи, до яких ви можете приєднатися. Ця технологія аналізує історичні дані про те, що вони полюбили

інші або вмісту, схожих на тих, які тобі сподобалися Користувач, для того, щоб підготувати рекомендації або додати їх канал Новини.

Крім того, технологія ML може використовуватися в інтернет-магазинах для рекомендацій по продуктам на основі попередніх покупок, пошуків та інших дій, виконуваних аналогічними користувачами.

Дуже важливою областю застосування технології ML в наші дні є розпізнавання зображень. З її допомогою платформи соціальних мереж дозволяють відзначати людей на фотографіях. Поліція використовує його для пошуку підозрюваних на фотографіях або відеозаписах. З незліченними камерами, встановленими в аеропортах, магазинах і з дверними дзвінками, ви можете виявити багатьох злочинців або подивитися, куди вони пішли.

Ще однією важливою областю застосування мл є медична діагностика. Після інциденту, такого як серцевий напад, ви можете переглянути дані, щоб виявити сигнали, які були пропущені. Для системи, використовуваної лікарів або лікарні можна вводити Історичні, медичні дані, щоб виявити зв'язок між вступними даними (поведінка, результат дослідження, ознака) і вихідними (наприклад, інфаркт). Коли в майбутньому лікар введе свої замітки і результати досліджень в систему, алгоритм зможе виявляти симптоми серцевого нападу набагато ефективніше, ніж це можуть зробити люди, що дозволить здійснювати профілактичні заходи.

Наступним прикладом використання технології ML є переклад контенту веб-сторінок і додатків на мобільні платформи. Деякі програми працюють краще, а інші гірше, що є результатом застосування різних моделей, методів і алгоритмів.В даний час технологія ML являє собою хліб насущний в системах банківських і кредитних карт. Виявлення деяких ознак шахрайства, яке для ML не представляє ніяких проблем, людині буде потрібно дуже багато часу або буде повністю поза його досяжності. Аналіз і маркування величезної кількості транзакцій (справедливих або помилкових) в майбутньому можуть дозволити

виявляти шахрайство в окремих операціях. Ідеальним типом машинного навчання (ML) для цієї мети є data mining.

Data mining-це тип машинного навчання, який включає в себе аналіз даних для складання прогнозів або виявлення шаблонів у великих наборах даних. Термін трохи вводить в оману, тому що ця технологія зовсім не полягає в тому, щоб прочісувати кого-небудь, будь то злочинець або співробітник, дані, щоб знайти щось корисне. Насправді це процес виявлення в даних шаблонів, які можуть стати в нагоді при прийнятті рішень в майбутньому.

Візьмемо, наприклад, компанію кредитних карт. Якщо у вас є кредитна картка, то ваш банк напевно хоч раз у житті інформував вас про підозрілу діяльність, пов'язану з цією карткою. Як банк виявив це так швидко і миттєво надіслав вам попередження? Цей тип захисту від шахрайства забезпечується тільки постійно працюючими алгоритмами Data mining. На початку 2020 року тільки в США налічувалося понад 1,1 трлн кредитних карт. Проведені з їх використанням транзакції генерують величезну кількість даних, використовуваних для видобутку даних, пошуку шаблонів і навчання виявленню підозрілої поведінки в майбутньому.

Deep learning (глибоке навчання) - це специфічний тип ML, заснований на нейронних мережах. Нейронна мережа імітує те, як працюють нейрони в мозку людини, щоб приймати рішення або розуміти щось. Наприклад, шестирічна дитина здатна відрізнити по обличчю свою маму від людини, що охороняє пішохідний перехід, тому що його мозок дуже швидко аналізує багато деталей-колір волосся, риси обличчя, шрами і т. Машинне навчання відтворює ці процеси в формі глибокого навчання.

Нейронна мережа має від трьох до п'яти шарів — один вхідний, від одного до трьох прихованих шарів і один вихідний шар. Приховані шари приймають рішення, що дозволяють поступово прийти до вихідного шару або висновку.

Який колір волосся? Який колір очей? Ви бачите шрам? Коли шарів кілька сотень, ми маємо справу з глибоким навчанням.

## 2.3 Застосування машинного навчання

Машинне навчання продовжує розвиватися і знаходить нові практичні застосування. Кількість можливих застосувань надзвичайно велика і дозволяє передбачити, що в майбутньому кожен аспект техніки буде включати в себе деяку реалізацію алгоритмів машинного навчання. Наприклад,:

- програмне забезпечення для розпізнавання мови
- ❖ Автоматичний переклад
- ❖ розпізнавання людської мови
- ❖ диктувати комп'ютера
- ❖ голосові керовані користувальницькі інтерфейси
- ❖ голосова Автоматизація домашніх дій
- ❖ інтерактивні офіси обслуговування
- ❖ розробка роботів
- автоматична навігація і управління:
- ❖ керування транспортним засобом (ALVINN)
- ❖ пошук шляху в незнайомому середовищі
- ❖ управління космічним кораблем (NASA Remote Agent)

- ❖ автоматизація виробничих і гірничодобувних систем (промисловість, гірничодобувна промисловість)

- аналіз і класифікація даних:

- ❖ систематика астрономічних об'єктів (HACA Sky Survey)
- ❖ діагностика захворювань на основі симптомів
- ❖ виявлення відмивання грошей
- ❖ розпізнавання почерку на основі прикладів
- ❖ Класифікація даних за тематичними групами за критеріями
- ❖ наближення невідомої функції на основі зразків
- ❖ визначення функціональних залежностей в даних
- ❖ прогнозування тенденцій на фінансових ринках
- ❖ моделювання та розробка лікарської терапії
- ❖ виявлення і підрахунок бактеріальних колоній в чашці Петрі
- ❖ на основі мікро - і макроекономічних даних

## 2.4 Підходи машинного навчання

### Навчання дерев рішень

Навчання дерев рішень (англ.Decision Tree learning) використовує в якості прогностичної моделі дерево рішень, яке зіставляє спостереження предмета з висновками про цільове значення предмета.

### Навчання представлень

Деякі алгоритми навчання, в основному алгоритми навчання без вчителя, призначені для виявлення кращих уявлень введення, наданих під час навчання. Класичні приклади включають метод основних компонентів і кластерний аналіз. Алгоритми навчання поданню (англ. representation learning) часто намагаються зберегти інформацію в своїх вхідних даних, але конвертувати їх в шляху, який робить, що це зручно, дозволяючи відновити вхідних даних, що надходять з невідомого розподілу, яка видає дані, і при цьому зовсім не обов'язково точна для настройки, які навряд чи через поширення.

Багатозадачні алгоритми навчання намагаються зробити це за межами обмежень, щоб навчені уявлення мали низький розмір. Рідкісні алгоритми кодування намагаються зробити це за межами обмежень, щоб навчені уявлення були рідкісними (у них було багато нулів). Алгоритми навчання поліноміального підпростору призначені для навчання низькомірних уявлень безпосередньо з тензорних уявлень багатовимірних даних, не перетворюючи їх у вектори (багатовимірні). Алгоритми глибокого навчання відкривають кілька рівнів представлення або ієрархію ознак, в яких більш високі, більш абстрактні риси визначаються з точки зору ознак нижчого рівня (або генерують їх). Було виявлено, що інтелектуальна машина-це та, яка вивчає уявлення, яке розплутує фактори, що лежать в основі спостереження.

## Індуктивне логічне програмування

Індукційні логічного програмування (LP, Inductive Logic programming, ILP) - це підхід до навчання з використанням правил логічного програмування як універсального представлення вхідних прикладів поширення проектування і гіпотез. Шляхом кодування відомих базових знань і набору прикладів, представлених в якості логічної бази даних фактів, система ILP вивести гіпотетичну логічну програму, яка має наслідки для всіх позитивних і негативних прикладів. Пов'язаною областю є індикативне програмування, яке для гіпотез враховує всілякі мови програмування (а не тільки логічне програмування), такі як функціональні програми.

## Кластерування

Кластерний аналіз - це розподіл набору спостережень на підмножини (звані кластерами), так що спостереження в одному кластері подібні відповідно до визначеного заздалегідь встановленим критерієм або критеріями, в той час як спостереження, взяті з різних кластерів, різні. Різні методи кластеризації приймають різні припущення про структуру даних, часто визначаються деякі вимірювання подібності і оцінюватися, наприклад, на внутрішню єдність (схожість членів одного і того ж кластера) і виразність між різними кластерами. Інші методи засновані на розрахунковій щільності і зв'язності графа. Групування-це метод навчання без вчителя і загальний метод статистичного аналізу даних.

## Баєсові мережі

Bayesian network, belief network, directed acyclic graphical model -і мовірнісна графічна модель, що представляє набір випадкових величин і їх умовних значень за допомогою цільового ациклічного графа. Наприклад, мережа баєса може представляти ймовірнісні зв'язки між захворюванням і симптомами. Маючи симптоми, таку мережу можна використовувати для розрахунку ймовірності наявності різних захворювань. Існують ефективні алгоритми для виконання висновків і навчання.

## Навчання з підкріпленням

Навчання з посиленням (reinforcement learning) розглядає, як агент повинен виконувати дії в навколишньому середовищі таким чином, щоб максимізувати деяке уявлення про довгострокову нагороду. Алгоритми навчання підкріплення намагаються знайти правила, що відображають стану світу в діях, які агент повинен виконувати в цих станах. Навчання з посиленням відрізняється від навчання з учителем, що пара правильних входів / виходів, ніколи йому не зображується, а не зовсім оптимальні дії ніколи не бувають чітко коригується.

## Навчання подібностей та мір

У цій проблемі машина, яка навчається, надає кілька прикладів, які вважаються схожими, і пари менш схожих об'єктів. Він повинен вивчити функцію подібності (або функцію вимірювання відстані), яка може передбачити, чи схожі нові об'єкти. Це іноді використовується в системах рекомендацій.

## Навчання розріджених словників

У цьому методі дані представлені лінійною комбінацією основних функцій, і передбачається, що коефіцієнти рідкісні. Нехай x буде d-розмірними даними, A D-матрицею D На n, кожен стовпець якої представляє базову функцію. r-це коефіцієнт, що представляє x з використанням D. в цілому, передбачається, що n більше d, щоб дати свободу рідкісному уявленню.

Вивчення словникового запасу поряд з рідкісним поданням є строго NP-складним і також складним для приблизного рішення. Популярним евристичним методом викладання рідкісних словників є K-CPM.

Викладання рідкісних словників використовувалося в декількох контекстах. У класифікації завдання полягає в тому, щоб визначити, до яких класів відносяться раніше не бачені дані. Припустимо, що словник для кожного з класів вже побудований. Потім нові дані пов'язані з класом, в якому словник рідко представлений найкраще. Викладання рідкісних словників також використовувалося у відомих образах. Ключова ідея полягає в тому, що чистий зріз зображення рідко може бути представлений словником зображення, а шум не може.

## Системи навчання класифікації

Системи навчання класифікації - це сімейство алгоритмів машинного навчання, заснованих на правилах, які поєднують в собі компонент (наприклад, правило, алгоритм генетичний) з компонентом навчання (який виконує навчання з гідом, навчання з посиленням або спонтанне навчання). Вони намагаються визначити набір контекстних правил, які в сукупності зберігають і застосовують знання нерівномірно для прогнозування.

## Контрольоване Машинне навчання

Цей метод використовує відомі, встановлені та класифіковані набори даних для пошуку шаблонів. Це дозволяє розвинути раніше описану ідею використання зображень собак і кішок. У вас може бути величезний набір даних, що містить зображення тисяч тварин на мільйонах фотографій. Оскільки ці види відомі, їх можна згрупувати і маркувати, перш ніж вводити в контрольований алгоритм ML, щоб навчити його розпізнавати їх. Потім такий контрольований алгоритм порівнює те, що він отримує на вході, з тим, що з'являється на виході, і фотографію з міткою виду тварини. Зрештою він навчиться розпізнавати певний вид тварини на нових фотографіях, які йому "покажуть".

## Автоматичне Машинне навчання

Нерегульована алгоритми МЛ сьогодні, як СПАМ-фільтри. Перші такі фільтри визначаються адміністраторами шукали певні слова в повідомленнях електронної пошти, щоб розуміти СПАМ. Тепер це неможливо, тому алгоритми без нагляду добре працюють. Без нагляду алгоритм ml отримує електронні листи, які не були позначені, щоб почати пошук шаблонів. Коли він виявляє їх, він дізнається, як виглядає спам, і ідентифікує його у виробничому середовищі.

## Регресійна техніка

Використовуючи регресію, ви можете прогнозувати ціни на нерухомість або визначити оптимальну ціну Снігової лопати в Міннесоті в грудні. Регресія заснована на принципі, що, хоча ціни можуть коливатися, вони завжди повертаються до певного середнього значення. Незважаючи на те, що з часом ціни на житло зростають, є середня ціна, яка завжди буде повторюватися. Щоб виявити середнє значення, засноване на змінах цін з плином часу, ви можете

нанести його на графік. Червона лінія діаграми, що піднімається вгору, дозволяє формулювати прогнози на майбутнє.

## Класифікація

Класифікація використовується для групування даних у відомі категорії. Наприклад, ви можете шукати таких людей, які, ймовірно, стануть хорошими клієнтами (ті, хто завжди повертається і витрачає більше грошей), або вони, ймовірно, змінять місце покупки. Якщо за допомогою спогадів ви зможете знайти типові характеристики клієнтів, що належать до певної групи, ви зможете передбачити, в яку з них потрапить кожен новий клієнт. Це дозволить вам більш ефективно проводити маркетингові кампанії і, можливо, отримати дуже лояльного клієнта, який в іншому випадку пішов би кудись ще. Це хороший приклад ml контрольованого.

## Алгоритми лінійної

Алгоритми лінійної регресії визначають відносини, поміщаючи на графік незалежні і залежні змінні і намітивши пряму лінію, що позначає середнє або тренд. Словник Merriam-Webster визначає регресію як "функцію, яка повертає середнє значення випадкової змінної за умови, що одна або кілька незалежних змінних мають певні значення"."Це визначення також стосується логістичної регресії.

## Угруповання

На відміну від класифікації, метод кластеризації відноситься до категорії ML без нагляду. Під час групування система знайде спосіб розділити дані на групи, які ви не знаєте, як згрупувати. Цей тип ML відмінно підходить для аналізу медичних зображень і соціальних мереж, а також для пошуку аномалій. Google використовує групування для узагальнення, стиснення даних і збереження конфіденційності в продуктах, таких як відео на YouTube, додатки в Play Store і музичні треки.

## Виявлення аномалій

Виявлення аномалій використовується для пошуку незвичайних об'єктів, таких як чорна вівця в стаді. Знайти їх у величезній масі даних людині неможливо. З іншого боку, якщо дослідник вводить в алгоритм пошуку аномалій платіжні дані з медичної системи декількох лікарень, він знайде спосіб згрупувати їх. Це дозволяє йому виявляти аномалії, які є ознаками обману.

## Аналіз ринкової корзини

Логіка аналізу ринкової корзини дозволяє формулювати прогнози. Простий приклад: якщо клієнти кладуть в свою корзину яловичий фарш, помідори і тако, можна очікувати, що вони дадуть ще сир і вершки. Ґрунтуючись на таких прогнозах, додаткові продажі можуть бути стимульовані шляхом надання клієнтам інтернет-магазинів привабливих пропозицій про покупку продуктів, про які вони можуть забути, або ви можете згрупувати продукти в магазині. За допомогою цієї техніки два професори з Массачусетського технологічного інституту виявили "вісники клапотя". Виявилося, що деякі клієнти люблять купувати продукти, які приречені на провал. Помітивши їх

присутність, ви дізнаєтеся, чи слід продовжувати продавати продукт і які рекламні заходи розгорнути, щоб максимізувати продажі у відповідній групі клієнтів.

### Часові ряди

Дані у вигляді часових рядів зазвичай збирають носяться нами на зап'ястях фітнес-Монітори. Вони записують кількість серцевих скорочень в хвилину, кількість кроків, які ви робите в хвилину або годину, а деякі з них вже вимірюють насичення. На основі таких даних можна передбачити, коли хтось буде бігати. Крім того, можна було б збирати дані машин, наприклад, про рівень вібрації, інтенсивності шуму і тиску, щоб передбачити збої.

# РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

## 3.1 Колаборативна фільтрація

Колаборативна/спільна фільтрація (Collaborative filtering — CF) – як правило, використовується рекомендаційними системами. Фільтрація спільної роботи має два значення: вузьке і більш загальне. Як правило, спільна фільтрація - процес фільтрації інформації або зразків методами, які включають співпрацю між декількома агентами, точками зору, джерелами даних і т. д. методи спільної фільтрації для різних типів даних, таких як Ідентифікація та моніторинг даних, що виникають при розвідці корисних копалин на великих площах; у фінансових даних, таких як установи фінансових послуг, які об'єднують декілька фінансових

джерел; або в електронній комерції та веб-додатках, вони зосереджені на користувацьких даних тощо.

У більш вузькому сенсі фільтрація спільної роботи служить для прогнозу, який використовує відомі переваги (оцінки) групи користувачів для прогнозування невідомих переваг іншого користувача. Основна передумова спільної фільтрації полягає в наступному: ті, хто в минулому оцінював будь-які предмети однаково, схильні давати аналогічні оцінки іншим предметам і в майбутньому. Наприклад, завдяки спільній фільтрації музичний додаток може передбачити, яку музику любить Користувач, маючи неповний список його переваг (симпатій і антипатій). Прогнози розробляються індивідуально для кожного користувача, хоча використовувана інформація збирається від декількох учасників. Це відрізняє фільтрацію співпраці від більш простого підходу, дає усереднений результат для кожного об'єкта, що цікавить, наприклад, на основі кількості голосів, поданих за нього.

Системи спільної фільтрації зазвичай використовують двоступеневу схему:

1. Знайде тих, хто поділяє оцінки "активного" (прогнозованого) користувача.
2. Використовує оцінки людей, які думають аналогічно тим, які були знайдені на першому етапі, щоб розрахувати прогноз.

Вище алгоритм побудований щодо користувачів системи.

Існує також альтернативний алгоритм, який був винайдений в Amazon, побудований по відношенню до предметів (продуктів) в системі. Цей алгоритм включає в себе наступні кроки:

1. Будується матриця, яка визначає відносини між парами об'єктів, щоб знайти схожі об'єкти.
2. Використовується побудована матриця і інформацію про Користувача, ми будуємо прогнози його оцінок.

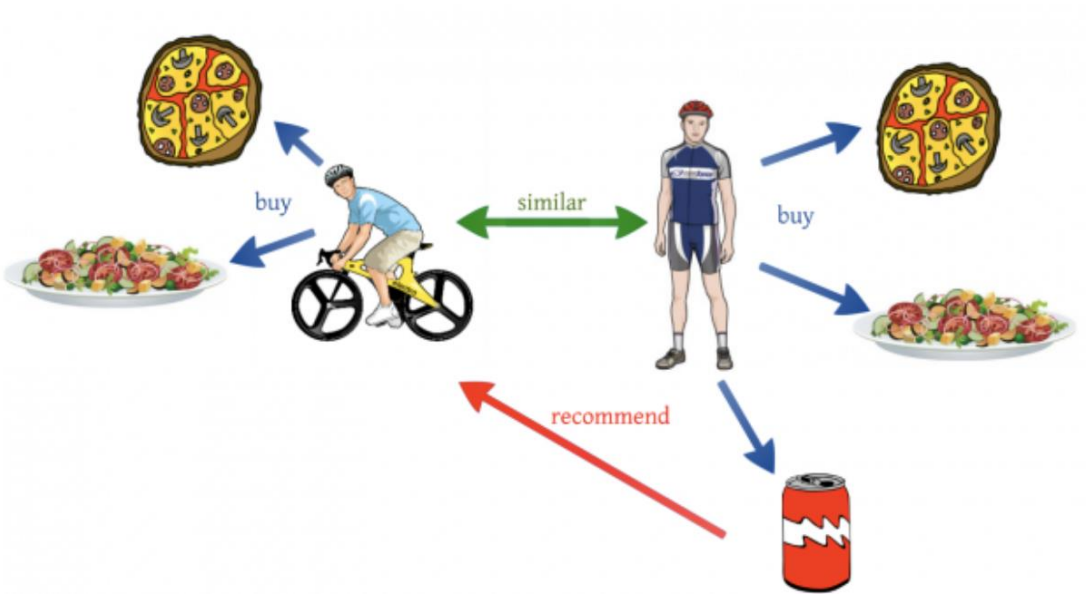


Рис. 3. Принцип колаборативної фільтрації

## Проблеми:

## Розрідженість даних

В цілому, більшість комерційних систем рекомендацій засновані на великій кількості даних (товарів), і більшість користувачів не оцінюють продукти. В результаті матриця суб'єкта-користувача дуже велика і рідкісна, що створює проблеми при обчисленні рекомендацій. Ця проблема особливо гостро ставиться до нових, тільки що створеним системам. Крім того, рідкість даних посилює проблему холодного запуску.

## Проблема холодного старту

Нові елементи або користувачі є великою проблемою для реферальних систем. Частково ця проблема допомагає вирішити підхід до аналізу контенту, оскільки він заснований не на рейтингах, а на атрибутах, що допомагає включати нові елементи в рекомендації користувачів. Проте вирішити проблему надання рекомендації новому користувачеві складніше.

## Масштабованість

Зі збільшенням кількості користувачів у системі виникає проблема масштабованості. Крім того, багато систем повинні негайно реагувати на онлайн-запити всіх користувачів, незалежно від їх історії покупок та рейтингу, що вимагає ще більшої масштабованості.

## Шахрайство

У системах рекомендацій, де кожен може оцінити, люди можуть оцінювати свої товари позитивно, а їхні конкуренти — погано. Крім того, реферальні системи почали справляти великий вплив на продажі та прибуток, оскільки вони широко використовувалися на комерційних сайтах. Це призводить до того, що недобросовісні постачальники намагаються несправедливо підвищити рейтинг своєї продукції та знизити рейтинг своїх конкурентів.

## Різноманітність

Спочатку було виявлено, що спільна фільтрація збільшує різноманітність, щоб дозволити користувачам відкривати нові продукти з різних жанрів. Однак деякі алгоритми, особливо засновані на продажах і рейтингах, створюють дуже складні умови для просування нових і маловідомих товарів, оскільки на зміну їм приходять популярні продукти, які вже давно присутні на ринку. Це, у свою чергу, лише посилює ефект «багаті стають багатшими» і призводить до меншої диференціації.

## Синонімія

Синонім тенденції однаковим і ідентичним об’єктам давати різні назви. Більшість реферальних систем не можуть виявити ці приховані зв’язки і тому сприймають ці теми як різні. Наприклад, «дитячі фільми» та «дитячі фільми» належать до одного жанру, але система трактує їх по-різному.

## Білі ворони

«Ворони» включають користувачів, чия думка не збігається з більшістю інших. Через унікальний смак їм нічого не можна порадити. Однак такі люди мають проблеми з отриманням порад у реальному житті, тому наразі немає пошуку вирішення цієї проблеми.

Підсумок, якщо ви шукаєте простий і ефективний спосіб поліпшити персоналізацію вашого інтернет-магазину, двигуни рекомендацій по продуктам є хорошими інвестиціями. Установка системи в інтернет-магазині незалежно від платформи проста і швидка і не вимагає знань в області програмування. Механізм рекомендацій надасть клієнтам кращий досвід покупок, показуючи їм продукти, на які вони інакше, ймовірно, не натрапили б.

# РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

## 4.1 Розробка мікросервісної частини

Для розробки веб-сервера була обрана платформа. NET Core через крос-платформенності, високій швидкості роботи, а також простоти розгортання в контейнерах. . NET Core також рекомендується для розробки додатків на основі мікросервісів.

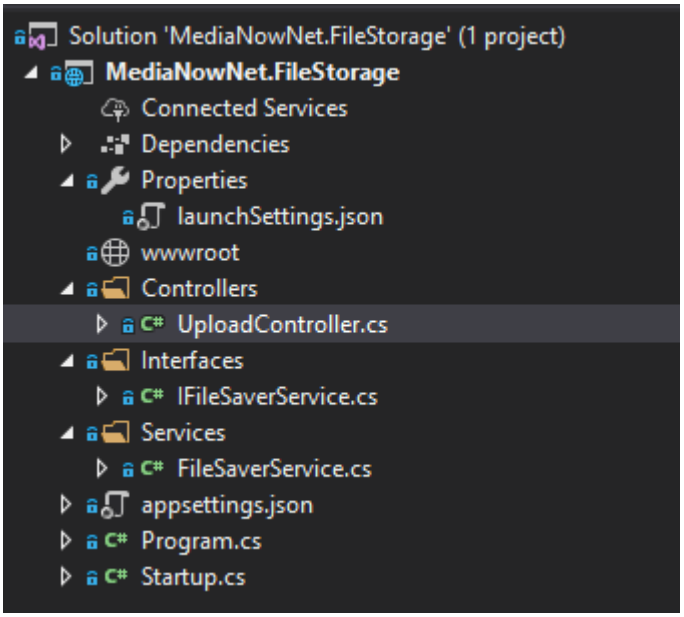


Рис.4. Структура файлового сервісу

Система потокового сервісу містить набір наступних мікросервісів:

1. Авторизаційний сервіс
2. Сервіс бібліотека з набором фільмів
3. Сервіс бібліотека з набором серіалів
4. Рекомендаційний сервіс
5. Файловий сервіс

Така грануляція  сервісів була обрана після поділу функціональних можливостей системи на певні функціональні контексти.

Взаємодія з сервісами відбувається через шлюз.

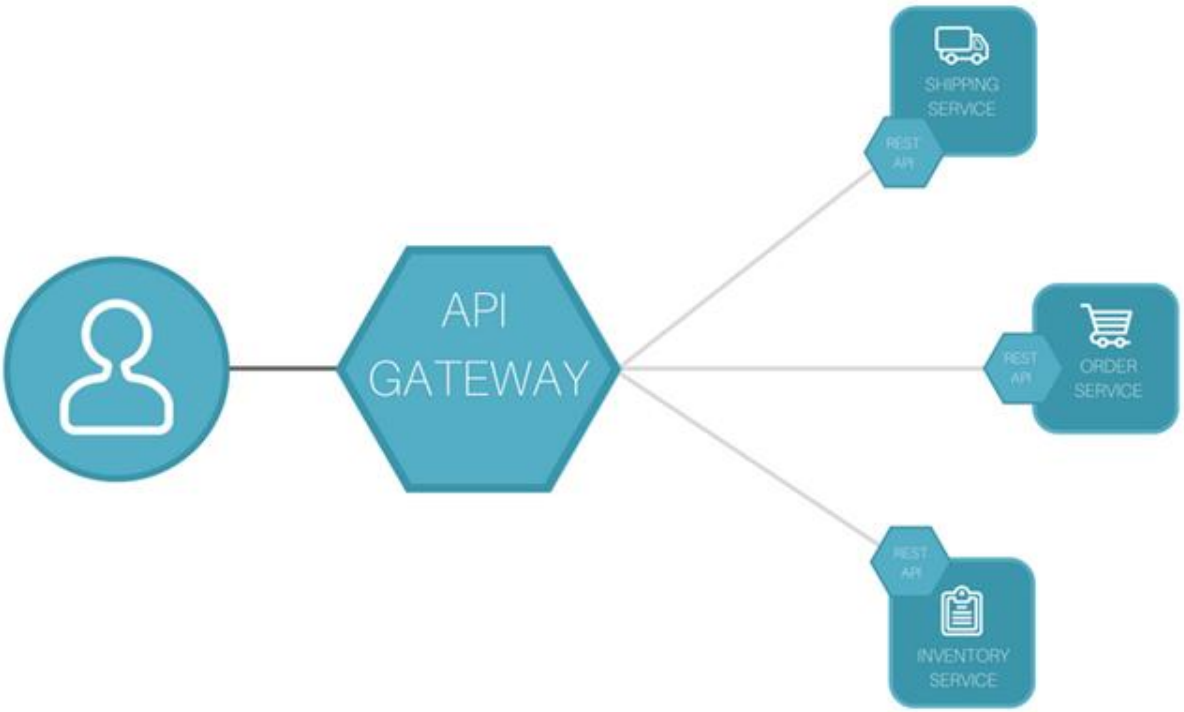


Рис.5. API Gateway

Такий принцип сумісності був обраний для уніфікації доступу до бекенда, оскільки кожна мікро-служба-це, по суті, окремий процес, окрема підсистема, яка може перебувати на абсолютно різних серверах. Завдання шлюзу полягає в тому, щоб спростити все це, тому що інтерфейсу потрібно буде знати тільки веб-адресу цього шлюзу.

Також для обміну повідомленнями між службами використовується система RabbitMQ. Якщо будь-яка з мікрослужб публікує повідомлення, воно поміщається в чергу, і всі передплатники цієї черги можуть завантажити його і реагувати певним чином.

Рівень бази даних використовує MS SQL Server, оскільки тип і специфіка даних моєї системи можуть бути представлені у вигляді реляційних таблиць

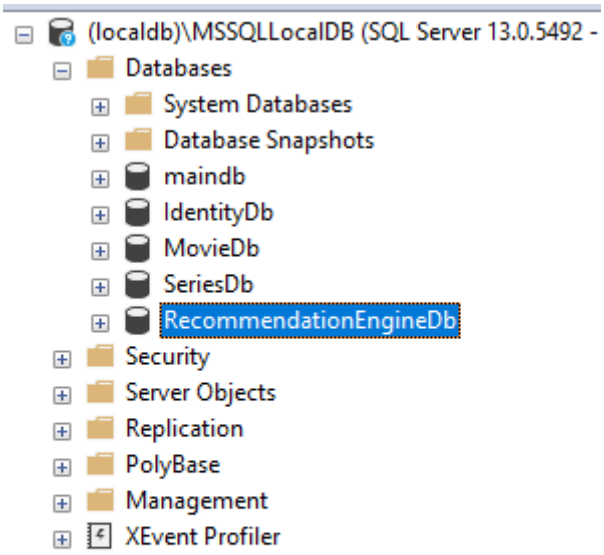


Рис.6. Структура баз даних на сервері

На наведеному вище малюнку показані існуючі бази даних для даної програми. За правилами мікросервісів бази діляться відповідним чином, кожна мікросервісів працює зі своїми даними.

```
var bus = Bus.Factory.CreateUsingRabbitMq(sbc =>
{
    var host = sbc.Host(new Uri("rabbitmq://localhost"), h =>
    {
        h.Username("guest");
        h.Password("guest");
    });

    sbc.ReceiveEndpoint(host, "test_queue", ep =>
    {
        ep.Handler<YourMessage>(context =>
        {
            return Console.Out.WriteLineAsync($"Received: {context.Message.Text}");
        });
    });
});

bus.Start(); // This is important!
```

Рис.7. Конфігурація обміну повідомленнями

```
public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
    services.AddAuthentication("Bearer")
        .AddJwtBearer("Bearer", options =>
        {
            options.Authority = "http://localhost:5000";
            options.RequireHttpsMetadata = false;
            options.Audience = "api1";
        });
}
```

Рис.8. Налаштування авторизації

## 4.2 Розробка клієнтської аплікації

Вибір бібліотеки для розробки клієнтського додатка був менш залежним від потреб проекту та більше від того, з якою платформою чи бібліотекою я мав справу більше, тому був обраний Angular 2+.

Як і бекенд, фронтенд також має свою архітектуру. Усе в проекті поділено на модулі:

- Feature Modules
- Core Modules
- Shared Modules

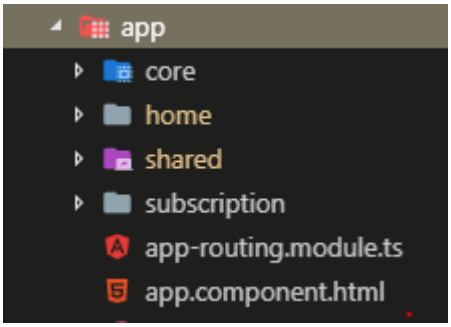


Рис.9. Модульна система

Для упрощення роботи з проєктом було використано Angular-cli. Це по факту набір консольних команд, який пришвидшує створення різноманітних компонентів проекту.

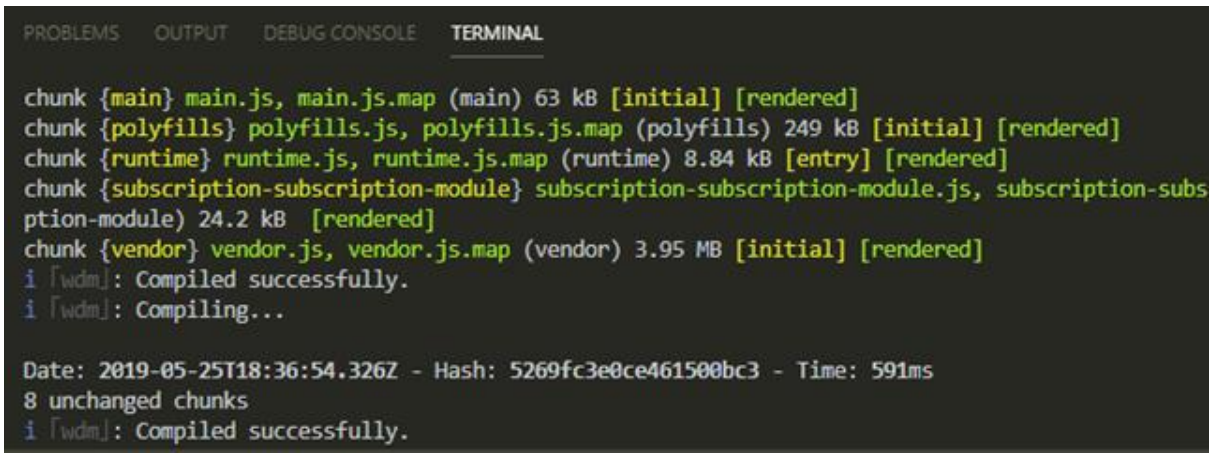


Рис.10. Результат компіляції за допомогою ng –serve

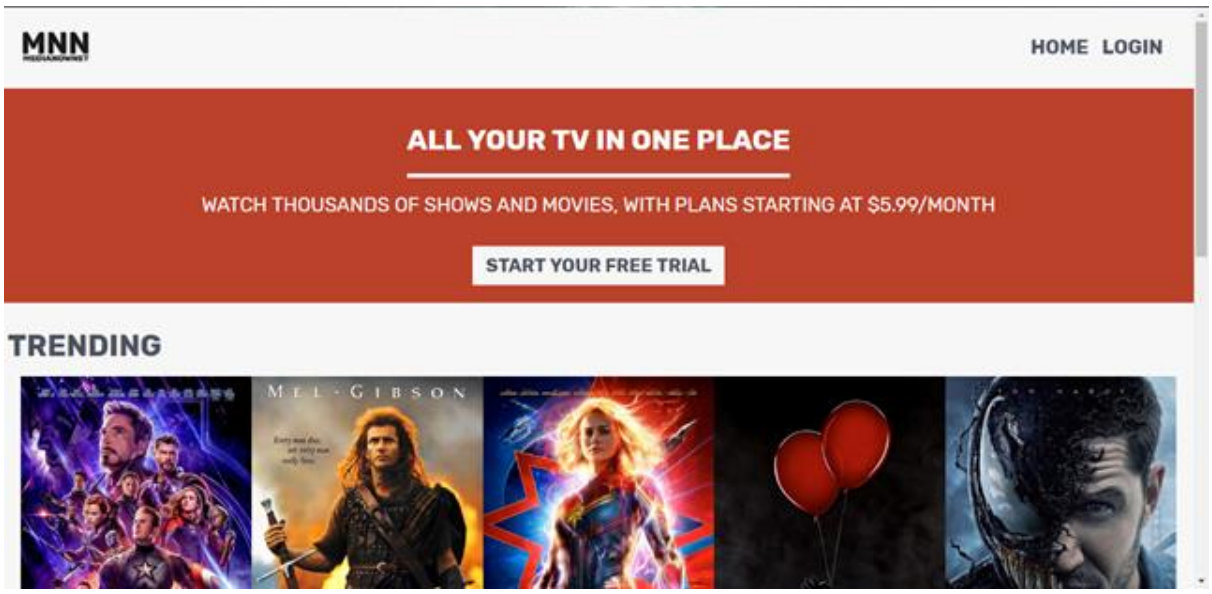


Рис.11. Вигляд веб-сайту

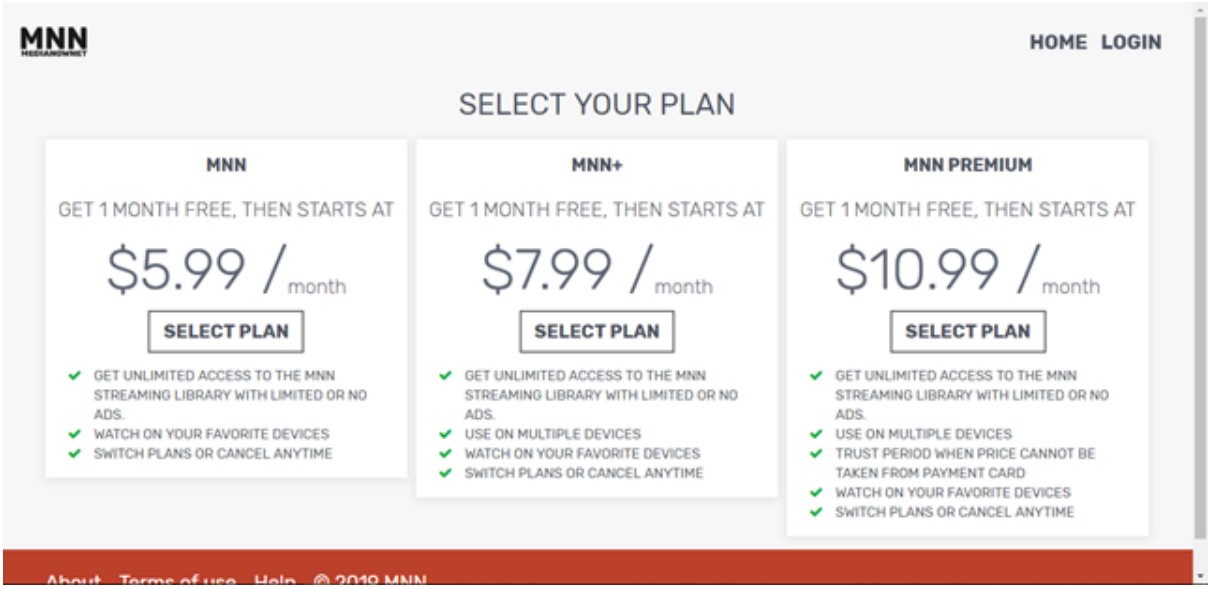


Рис.12.Тарифні плани

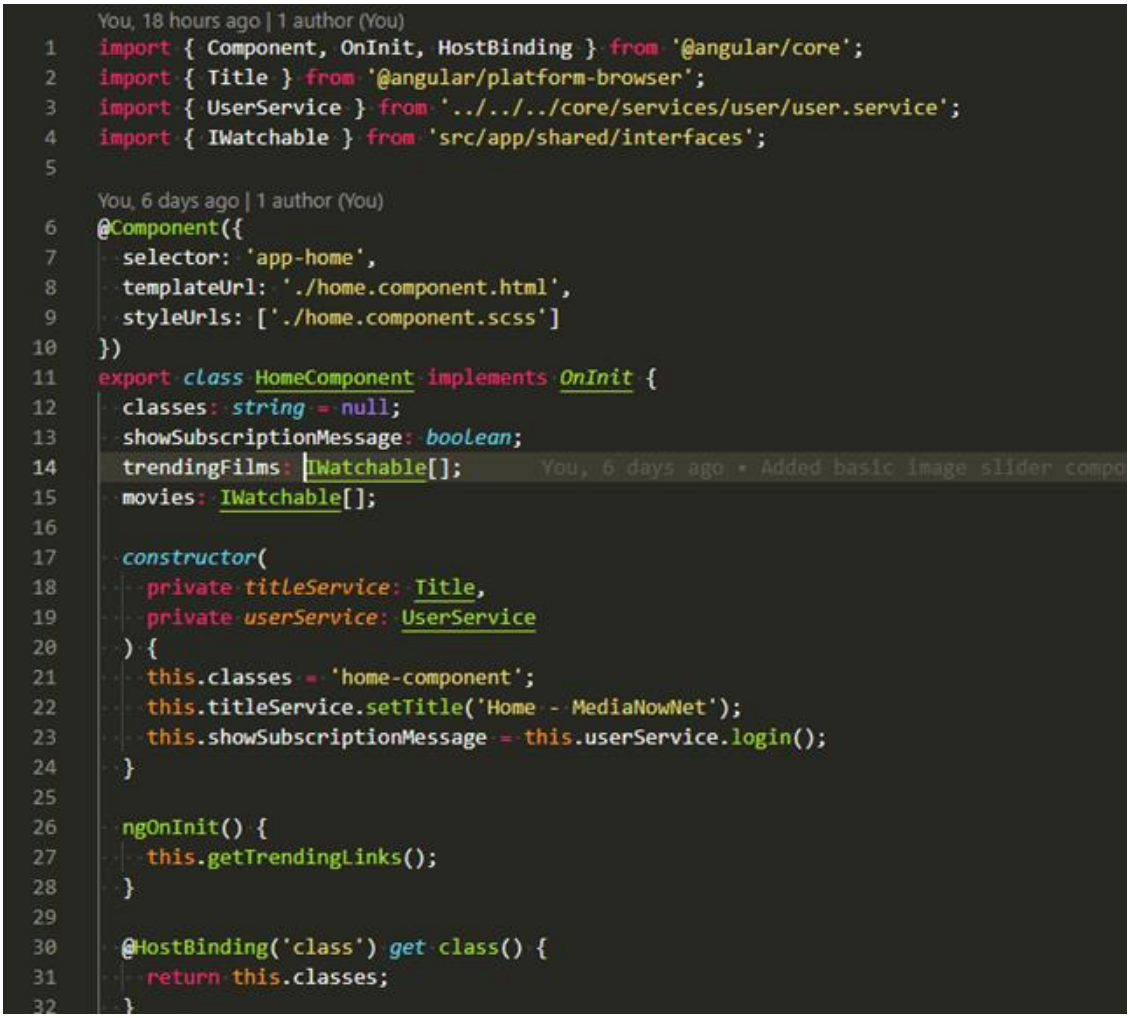


Рис.13. Приклад компоненти

Також з метою тестування фронтенду на різних пристроях, веб-сайт було розміщено на хмарному сервісу Heroku.

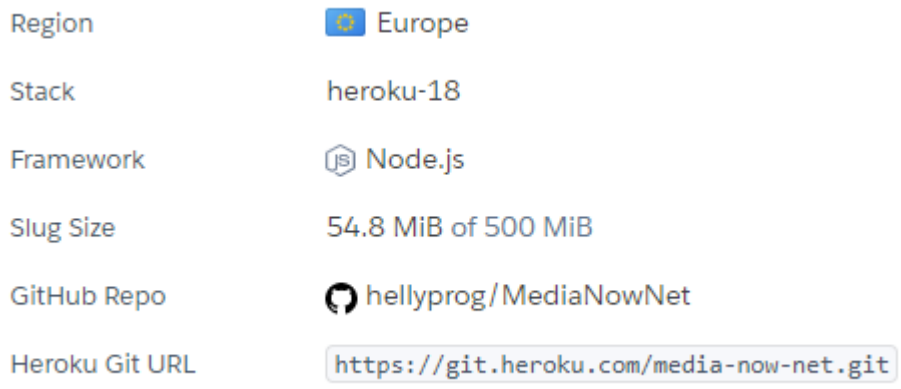


Рис.14. Дані про аплікацію з сервісу Heroku

Скрипт для розгортання фронтенду був написаний з використанням Node.js

```
const express = require('express');
const path = require('path');

const app = express();

// Serve only the static files form the dist directory
app.use(express.static(__dirname + '/dist/MediaNowNet'));

app.get('/*', function(req, res) {
  res.sendFile(path.join(__dirname+'/dist/MediaNowNet/index.html'));
});

// Start the app by listening on the default Heroku port
app.listen(process.env.PORT || 8080);
```

Рис.15. Скрипт для розгортання

## 4.3 Розгортання у контейнерах

Так як сервіси були розроблені на платформі .NET Core їх легко розміщувати в контейнерах. Для цього в проект потрібно додати **dockerfile.** Це спеціальний скрипт який вказує докеру як і куди потрібно перемістити результати компіляції бекенду.

Dockerfile

```
1  FROM microsoft/dotnet:2.1-aspnetcore-runtime AS base
2  WORKDIR /app
3  EXPOSE 80
4  EXPOSE 443
5
6  FROM microsoft/dotnet:2.1-sdk AS build
7  WORKDIR /src
8  COPY ["DockerSample/DockerSample.csproj", "DockerSample/"]
9  RUN dotnet restore "DockerSample/DockerSample.csproj"
10 COPY . .
11 WORKDIR "/src/DockerSample"
12 RUN dotnet build "DockerSample.csproj" -c Release -o /app
13
14 FROM build AS publish
15 RUN dotnet publish "DockerSample.csproj" -c Release -o /app
16
17 FROM base AS final
18 WORKDIR /app
19 COPY --from=publish /app .
20 ENTRYPOINT ["dotnet", "DockerSample.dll"]
```

Рис.16. dockerfile

Після додавання такого файла в проект треба прописати декілька команд, щоб сервіс запустився в контейнері.

```
PS E:\Software Development\Projects\DockerDotNetDevsSample1> docker run -
p 8080:80 sample1

Hosting environment: Production

Content root path: /app/src/DockerDotNetDevsSample1

Now listening on: http://+:80

Application started. Press Ctrl+C to shut down.
```

Підкресленою є команда яка вказує, що наш проект потрібно побудувати і запустити на локальному сервері на 8080 порті.

## 4.4 Створення моделі рекомендаційної системи потокового сервісу

Для виконання задачі машинного навчання потрібно виконати наступні кроки:

- Завантаження даних
- Побудова і тренування моделі
- Оцінка моделі
- Використання моделі

Також важливо правильно оцінити проблему, яку буде вирішувати завдання машинного навчання. В даному випадку завданням буде передбачення оцінки

певному фільму для того, щоб визначити чи бал оцінки достатньо високий для того, щоб його рекомендувати.

Першим кроком було завантажено підготовлені дані.

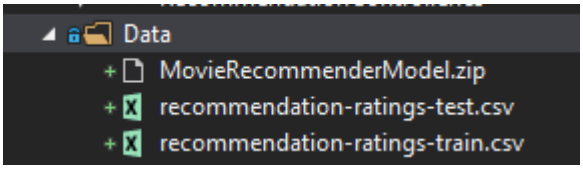


Рис.17. Файли з даними для алгоритму

Два файли в форматі .csv. Один файл використовуєтсья для тренування моделі, інший для тестування оцінки роботи моделі.

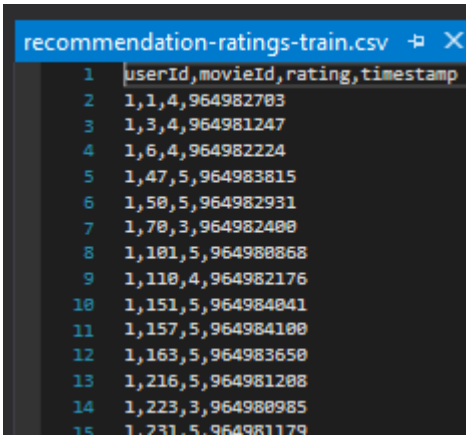


Рис.18. Вигляд тренувальних даних

Дані в файлі представлені зв’язком користувач – фільм – рейтинг.

В даному випадку критеріями по яких буде отриманий результат (features) - користувач та фільм, а результатом буде рейтинг (label).

Тренувальні алгоритми машинного навчання потребують даних у певному форматі, для цього використовуються трансформери. В ML.NET трансформери створюються за топомогою естіматорів.

```
1 reference
public static ITransformer BuildAndTrainModel(MLContext mlContext, IDataView trainingDataView)
{
    IEstimator<ITransformer> estimator = mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "userIdEncoded",
                                        .Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "movieId

    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = "userIdEncoded",
        MatrixRowIndexColumnName = "movieIdEncoded",
        LabelColumnName = "Label",
        NumberOfIterations = 40,
        ApproximationRank = 100
    };

    var trainerEstimator = estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));
    ITransformer model = trainerEstimator.Fit(trainingDataView);

    return model;
}
```

Рис.19. Процес побудови програмної моделі

Для поставленої задачі був використаний естіматор матричної факторизації. Цей алгоритм використовує метод колаборативного фільтрування, який був описаний раніше.

Результатом тренування моделі буде наступний вивід:

```
=============== Training the model ===============
iter      tr_rmse          obj
   0       1.5403   3.1262e+05
   1       0.9221   1.6030e+05
   2       0.8687   1.5046e+05
   3       0.8416   1.4584e+05
   4       0.8142   1.4209e+05
   5       0.7849   1.3907e+05
   6       0.7544   1.3594e+05
   7       0.7266   1.3361e+05
   8       0.6987   1.3110e+05
   9       0.6751   1.2948e+05
  10       0.6530   1.2766e+05
  11       0.6350   1.2644e+05
  12       0.6197   1.2541e+05
  13       0.6067   1.2470e+05
  14       0.5953   1.2382e+05
  15       0.5871   1.2342e+05
  16       0.5781   1.2279e+05
  17       0.5713   1.2240e+05
  18       0.5660   1.2230e+05
  19       0.5592   1.2179e+05
=============== Evaluating the model ===============
Rms: 0.994051469730769
RSquared: 0.412556298844873
```

Рис.20. Результат оцінки моделі

В цьому виводі 20 ітерацій, на кожній ітерації величина помилки зменшується і прямує ближче до нуля. RMS використовується для оцінки різниці між передбаченням моделі та тестовими даними, чим нижчий цей показник, тим краще. Rsquared визначає наскільки дані підходять для моделі, варіюється від 0 до 1, чим ближчий цей показник до 1 тим краще.

В результаті тренування моделі отримується .zip файл який містить цю модель і його можна використовувати в інших проектах для передбачень.

Для представлення результатів було взято випадкового користувача з ID = 6, а також фільм з MovieID = 10.

З поточним набором  даних отримано наступну оцінку:

```
{
    "label": 0,
    "score": 3.749414
}
```


Рис.21. Оцінка фільму для випадкового користувача

При цьому максимальна оцінка 5 балів

Далі було збільшено тренувальний набір даних користувачами, які поставили високий бал фільму з MovieID = 10.

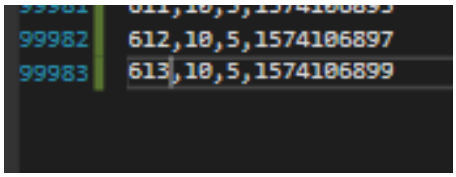


Рис.22. Нові дані у тренувальному наборі

Було додано ще двох користувачів, які оцінили фільм максимальним балом.

Після тренування нашого алгоритму отримано наступну оцінку:


```
{
    "label": 0,
    "score": 3.78768
}
```


Рис.23. Оцінка системою з новими даними

Як бачимо, навіть 2 оцінки підняли рекомендований рейтинг фільму для даного користувача.

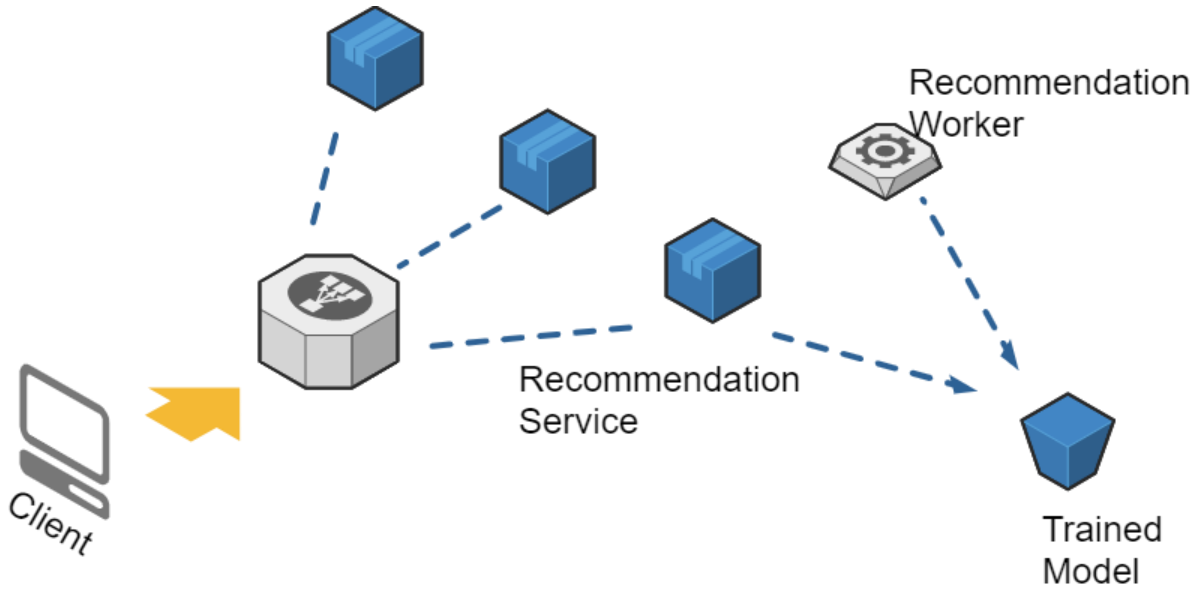


Рис.24. Місце рекомендацій у системі

Так як результатом тренування моделі є zip файл, який використовується згодом в ML.NET для беспосереднього виконання задач, то потрібно мати змогу цей файл перегенерувати. Для цього може бути використано окремий сервіс, який буде запускатись автоматично з певною періодичністю і тренуватиме модель на основі свіжих даних.

# РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

## 5.1 Опис ідеї проекту

*Зміст ідеї* полягає в створенні мікросервісного веб сайту для пошуку фільмів і окремий сервіс, який використовує алгоритм “колаборативної фільтрації” та надає рекомендації кіно для користувача.

*Можливі напрямки застосування*. Сайт можна використовувати для пошуку фільмів, оцінки якості фільмів. Даний сайт зможе допомогти користувачеві вибрати фільм за допомогою рекомендаційного механізму.

*Основні вигоди, що може отримати користувач*. На сьогоднішній день, кіноіндустрія включає велетенську кількість фільмів, звичайний користувач не може бути в курсі всіх новинок або найкращих фільмів певного жанру. Даний сайт допоможе зняти тяжкий процес пошуку фільму з користувача. З бізнес точки зору, такий сайт буде безплатний для користувачів, а прибуток отримуватиметься від реклами.

*Відмінність від існуючих аналогів.* Сайт буде мати обов’язково український інтерфейс. А найголовніше це зручний користувацький інтерфейс, тільки в такому випадку стартап матиме змогу конкурувати на ринку.

## 5.1 Розроблення ринкової стратегії

Дана веб система розрахована на користувачів любого віку, достатнього щоб користуватися браузером. Для системи не важливі стать чи інші характеристики користувача. Тому для такої системи добре піддійте використання масового маркетингу Масовий маркетинг-це маркетингова стратегія, при якій фірма вирішує ігнорувати відмінності в сегментах ринку і залучати весь ринок однією пропозицією або однією стратегією, яка підтримує ідею поширення повідомлення, яке охопить якомога більше число людей. Традиційно Масовий маркетинг був зосереджений на радіо, телебаченні та газетах, оскільки засоби масової інформації використовувалися для охоплення цієї широкої аудиторії. Охоплюючи якомога більшу аудиторію, ви максимально знайомитеся з продуктом, і теоретично це буде безпосередньо корелювати з великою кількістю продажів або покупок продукту.

## 5.2 Розроблення маркетингової програми

Так як діяльність системи направлена на споживачів різних категорій, то і просування, і розвиток буде направлений саме на різні сегменти. Це пов’язано перш за все з тим, що жанрів в кіно є велика кількість. Деякі жанри підходять для всіх глядачів, а декотрі мають вікове обмеження. В залежності від віку та вибраних жанрів буде показуватися відповідна реклама. Такий підхід допоможе отримати прибуток від рекламодавців і не буде відлякувати користувачів від користування системою. Ще можна розвинути в майбутньому можливість стрімінгу фільмів за різними тарифними планами

## 5.3 Вимоги до технічного та програмного забезпечення

Для даної системи вибрана сучасна архітектура, а саме мікросервісна. Даний вибір для бекенду пояснюється тим що:

- Є можливість реалізування на різних мовах програмування, СБД, та ін.
- Кожен сервіс гнучкий, легко компонується та взаємодіє з іншими сервісами, стійкий до відмов та вразливостей, функціонально мінімальний та завершений.
- Немає залежності від конкретної технології та одного постачальника.
- Простота проектування і внесення змін в окремі елементи.
- Можна маштабувати мікросервісну архітектуру в доволі простий спосіб, використовуючи хмарне середовище.

Для веб частини вибрано ангуляр, тому що це популярна технологія. Яка розвивається і є досить простою у використанні.

# Висновки

В результаті роботи був розроблений веб-сайт, заснований на архітектурі мікросервісів, а також мікросервіс для надання рекомендацій. Під час розробки були проаналізовані плюси і мінуси мікросервісів. Серед переваг використання цієї архітектури можна виділити незалежність розробки і масштабування, простоту заміни однієї реалізації іншої. Мікросервіси також мають деякі недоліки: дублювання коду, проблема розподілених транзакцій, невідповідність даних. При створенні двигуна рекомендацій, варіанти алгоритмів були вивчені для нього. З них був обраний найбільш оптимальний для набору даних, який використовує розроблену систему, а саме алгоритм матричної факторизації, в якості компонента спільної фільтрації.

# Список використаної літератури


1. MSDN - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://msdn.Microsoft.com/library
2. .NET Microvervices for containerized Appliccations by Microsoft– 331с.
3. Metanit - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://metanit.com/
4. Microservices Patterns - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://microservices.io/patterns/index.html
5. C# in a nutshell by J.Albahari– 1131с.
6. Docker Official Website - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://docs.docker.com/engine/examples/dotnetcore/
7. Angular Documentation - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://angular.io
8. ML.NET Documentation - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://dotnet.Microsoft.com/apps/machinelearning-ai/ml-dotnet
9. Wikipedia - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://en.wikipedia.org/wiki/Collaborative_filtering
10. Cyptobook - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://cryptobook.nakov.com/
11. Telegra- [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://telegra.ph/docker-04-06
12. Docker - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу: https://www.stevejgordon.co.uk/docker-for-dotnet-developers-part-2
13. Wikipedia - [Електронний ресурс]:[Веб-сайт]. – Електронні дані. – Режим доступу:https://uk.m.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F

# Додатки

## Додаток 1. Код програми

**File Storage Service**

```
namespace MediaNowNet.FileStorage.Interfaces
{
    public interface IFileSaverService
    {
        ExecutionResult SaveFile(string webRootPath, IFormFile file);
        ExecutionResult DeleteFile(string webRootPath, string link);
    }
}

namespace MediaNowNet.FileStorage
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);

            services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
            services.AddTransient<IFileSaverService, FileSaverService>();
        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            var cachePeriod = "604800";

            if (env.IsDevelopment())
            {
                cachePeriod = "600";
                app.UseDeveloperExceptionPage();
            }

            app.UseStaticFiles(new StaticFileOptions
            {
                OnPrepareResponse = ctx =>
                {
                    ctx.Context.Response.Headers.Append("Cache-Control", $"public, max-
age={cachePeriod}");
                }
            });

            app.UseMvc();
        }
    }
}
```

```
namespace MediaNowNet.FileStorage.Services
{
    public class FileSaverService : IFileSaverService
    {
        private readonly IHttpContextAccessor _httpContextAccessor;

        public FileSaverService(IHttpContextAccessor httpContextAccessor)
        {
            _httpContextAccessor = httpContextAccessor;
        }

        public ExecutionResult DeleteFile(string webRootPath, string link)
        {
            Log.Information($"FileSaverService -> SaveFile. webRootPath: {webRootPath},
fileName: {link}");

            try
            {
                Uri uri = new Uri(link);
                var fileName = uri.Segments.Last();
                string fullPath = $@"{webRootPath}\{fileName}";

                if (File.Exists(fullPath))
                {
                    File.Delete(fullPath);
                    return new ExecutionResult { Success = true };
                }

                return new ExecutionResult { Success = false, ErrorMessage = "File cannot be
saved" };
            }
            catch (Exception ex)
            {
                Log.Error(ex, "Saving failed: FileSaverService -> DeleteFile");
                return new ExecutionResult { Success = false, ErrorMessage = ex.Message };
            }
        }
```

```
        public ExecutionResult<string> SaveFile(string webRootPath, IFormFile file)
        {
            Log.Information($"FileSaverService -> SaveFile. webRootPath: {webRootPath},
file: {JsonConvert.SerializeObject(file)}");

            try
            {
                var fileName =
$"{Guid.NewGuid().ToString()}{Path.GetExtension(file.FileName)}";
                var path = $@"{webRootPath}\{fileName}";

                using (FileStream fs = File.Create(path))
                {
                    file.CopyTo(fs);
                    fs.Flush();
                }

                var request = _httpContextAccessor.HttpContext.Request;
                var savedFilePath = $@"{request.Scheme}:\\{request.Host}\{fileName}";

                return new ExecutionResult<string> { Success = true, Data = savedFilePath };
            }
            catch (Exception ex)
            {
                Log.Error(ex, "Saving failed: FileSaverService -> SaveFile");
                return new ExecutionResult<string> { Success = false, ErrorMessage =
ex.Message };
            }
        }
    }
}
```

```
namespace MediaNowNet.FileStorage.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class FileController : ControllerBase
    {
        private readonly IHostingEnvironment _environment;
        private readonly IFileSaverService _fileSaverService;

        public FileController(IHostingEnvironment environment, IFileSaverService
fileSaverService)
        {
            _environment = environment;
            _fileSaverService = fileSaverService;
        }

        [HttpPost]
        public IActionResult Upload(IFormFile file)
        {
            Log.Information($"UploadController -> Post. File:
{JsonConvert.SerializeObject(file)}");

            if (file != null)
            {
                var result = _fileSaverService.SaveFile(_environment.WebRootPath, file);

                return result.Success
                    ? StatusCode(StatusCodes.Status200OK, result)
                    : StatusCode(StatusCodes.Status500InternalServerError, result);
            }

            return StatusCode(StatusCodes.Status500InternalServerError, new ExecutionResult
{ Success = false });
        }

        [HttpDelete]
        public IActionResult Delete(string link)
        {
            Log.Information($"UploadController -> Delete. FileName: {link}");

            var result = _fileSaverService.DeleteFile(_environment.WebRootPath, link);

            return result.Success
                ? StatusCode(StatusCodes.Status204NoContent)
                : StatusCode(StatusCodes.Status500InternalServerError);
        }
    }
}
```

```
namespace MediaNowNet.FileStorage
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var configuration = GetConfiguration();
            Log.Logger = GetLoggerConfiguration(configuration);

            try
            {
                Log.Information("Starting web host");
                CreateWebHostBuilder(configuration, args).Build().Run();
            }
            catch (Exception ex)
            {
                Log.Fatal(ex, "Host terminated unexpectedly");
            }
            finally
            {
                Log.CloseAndFlush();
            }
        }

        public static IWebHostBuilder CreateWebHostBuilder(IConfiguration configuration,
string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>()
                .UseConfiguration(configuration)
                .UseSerilog();

        public static IConfiguration GetConfiguration() => new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
                                                                .Build();

        public static Serilog.ILogger GetLoggerConfiguration(IConfiguration configuration)
=> new LoggerConfiguration()

.ReadFrom.Configuration(configuration)

.CreateLogger();
    }
}
```

**Recommendation engine**

```
namespace MediaNowNet.RecommendationEngine
{
       public class MovieRatingPrediction
       {
              public float Label { get; set; }
              public float Score { get; set; }
       }
}
```

```
namespace MediaNowNet.RecommendationEngine.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class RecommendationController : ControllerBase
    {
            [HttpGet]
            public async Task<IActionResult> GetRecommendation()
            {
                    var mlContext = new MLContext();
                    (var trainingDataView, var testDataView) = LoadData(mlContext);
                    var model = BuildAndTrainModel(mlContext, trainingDataView);
                    EvaluateModel(mlContext, testDataView, model);
                    var prediction = UseModelForSinglePrediction(mlContext, model);
                    SaveModel(mlContext, trainingDataView.Schema, model);

                    return Ok(prediction);
            }

            private (IDataView training, IDataView test) LoadData(MLContext mlContext)
            {
                    var trainingDataPath = Path.Combine(Environment.CurrentDirectory,
"Data", "recommendation-ratings-train.csv");
                    var testDataPath = Path.Combine(Environment.CurrentDirectory, "Data",
"recommendation-ratings-test.csv");

                    var trainingDataView =
mlContext.Data.LoadFromTextFile<MovieRating>(trainingDataPath, hasHeader: true,
separatorChar: ',');
                    var testDataView =
mlContext.Data.LoadFromTextFile<MovieRating>(testDataPath, hasHeader: true, separatorChar:
',');

                    return (trainingDataView, testDataView);
            }
```

```
public static ITransformer BuildAndTrainModel(MLContext mlContext, IDataView
trainingDataView)
            {
                  IEstimator<ITransformer> estimator =
mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "userIdEncoded",
inputColumnName: "UserId")

      .Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName:
"movieIdEncoded", inputColumnName: "MovieId"));

                  var options = new MatrixFactorizationTrainer.Options
                  {
                        MatrixColumnIndexColumnName = "userIdEncoded",
                        MatrixRowIndexColumnName = "movieIdEncoded",
                        LabelColumnName = "Label",
                        NumberOfIterations = 40,
                        ApproximationRank = 100
                  };

                  var trainerEstimator =
estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));
                  ITransformer model = trainerEstimator.Fit(trainingDataView);

                  return model;
            }


            public static void EvaluateModel(MLContext mlContext, IDataView testDataView,
ITransformer model)
            {
                  var prediction = model.Transform(testDataView);
                  var metrics = mlContext.Regression.Evaluate(prediction,
labelColumnName: "Label", scoreColumnName: "Score");
                  Console.WriteLine("Root Mean Squared Error : " +
metrics.RootMeanSquaredError.ToString());
                  Console.WriteLine("RSquared: " + metrics.RSquared.ToString());
            }

            public static MovieRatingPrediction UseModelForSinglePrediction(MLContext
mlContext, ITransformer model)
            {
                  var predictionEngine =
mlContext.Model.CreatePredictionEngine<MovieRating, MovieRatingPrediction>(model);
                  var testInput = new MovieRating { UserId = 6, MovieId = 10 };

                  return predictionEngine.Predict(testInput);
            }
```

```
                public static void SaveModel(MLContext mlContext, DataViewSchema
    trainingDataViewSchema, ITransformer model)
                {
                        var modelPath = Path.Combine(Environment.CurrentDirectory, "Data",
    "MovieRecommenderModel.zip");
                        mlContext.Model.Save(model, trainingDataViewSchema, modelPath);
                }
        }
    }

    namespace MediaNowNet.RecommendationEngine
    {
        public class MovieRating
        {
            [LoadColumn(0)]
            public float UserId { get; set; }
            [LoadColumn(1)]
            public float MovieId { get; set; }
            [LoadColumn(2)]
            public float Label { get; set; }
        }
    }
```

**Angular app**

```
import { CoreModule } from './core';

import { HttpModule } from '@angular/http';

import { CommonModule } from '@angular/common';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { FormsModule } from '@angular/forms';

import { ToastrModule } from 'ngx-toastr';
```

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'MediaNowNet';
}
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';


import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { SharedModule } from './shared';
```

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    AppRoutingModule,
    BrowserModule,
    CoreModule,
    CommonModule,
    HttpModule,
    SharedModule,
    BrowserAnimationsModule,
    ToastrModule.forRoot(),
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
```

```
})
export class AppModule { }
<div class="main-container">
  <app-header class="header"></app-header>
  <div class="content">
    <router-outlet></router-outlet>
  </div>
  <app-footer></app-footer>
</div>
```

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```