Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Розробка додатку для рекомендацій закладів (ресторанів/кафе) Києва

Текстова частина до курсової роботи
за спеціальністю „Комп'ютерні науки"

Керівник курсової роботи
ст. викладач Борозенний С.О.
(прізвище та ініціали)

_____
(підпис)
"____" _____ 2022 р.

Виконав студент
ВакуленкоС.С.
(прізвище та ініціали)

"____" _____ 2022 р.

Київ 2022

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
Доцент., к. ф.-м. н. О.П.Жижерун
_____
(підпис)
„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Вакуленко Сергію Сергійовичу факультету інформатики 4 курсу

ТЕМА: Розробка додатку для рекомендацій закладів (ресторанів/кафе) Києва

Зміст ТЧ
1.Індивідуальне завдання
2.Календарний план
3.Вступ
4.Постановка завдання курсової роботи
5.Теоретичні відомості
6.Огляд створеного додатку та аналіз результатів
7.Висновки
8.Список використаних джерел

Дата видачі „___” _____ 2022 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема: Розробка додатку для рекомендацій закладів (ресторанів/кафе) Києва

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Приміт ка
1.	Отримання завдання на курсову роботу.	17.09.2021	
2.	Огляд літератури за темою роботи.	21.11.2021	
3.	Виконання аналізу актуальності теми	13.02.2022	
4.	Розробка програмного застосунку	05.05.2022	
5.	Написання пояснювальної роботи.	15.05.2022	
6.	Попередня демонстрація проекту науковому керівнику	18.05.2022	

Студент _____Вакуленко С.С,_____ _____

Керівник _____Борозенний С.О_____ _____

“_____”_____

ЗМІСТ

ВСТУП

Життя людини складається з постійного вибору. Яку книжку прочитати? Що купити? Куди піти? Такі, та багато схожих запитань людина ставила собі і десятки, сотні років тому, але вибір був значно менший, що давало можливість легше прийняти рішення. Люди робили покупки в звичайних магазинах, у яких кількість доступних товарів була обмежена.

Зараз, завдяки шаленому науково-технічному прогресу та поширенню інтернету, людина все частіше постає у ситуації вибору. Мати вибір приємно, однак виникає нова проблема - стало неймовірно важко приймати рішення.

Система рекомендацій – це система, яка здатна передбачити майбутні переваги набору елементів для користувача та рекомендувати найпопулярніші та ті елементи, що підходять найкраще.

Рекомендаційні системи розробляються завдяки ML. Ми звикли вважати, що якщо необхідно використовувати ML, то мовою розробки має бути Python. Але, крім Python, є безліч чудових мов програмування, які можна використовувати для написання таких застосунків. Я вирішив довести, що C# також хороша мова, для створення рекомендаційної системи.

Мета курсової - розібратись детальніше з різними рекомендаційними системами, їх перевагами та недоліками, важкістю реалізації та особливістю використання. Розглянути мову C#, як мову для створення рекомендаційного застосунку.

Завданням курсової роботи є розробка додатку, що буде допомагати користувачеві прийняти вибір, який заклад харчування відвідати.

Застосунок розроблений на мові C# з використанням фреймворку для машинного навчання ML.NET. В якості сервера, використовувався MS SQL Server.

Написання коду здійснювалось у Microsoft Visual Studio 2022, а для керування, налаштування та адміністрування всіх компонентів SQL Server використовувався SQL Server Management Studio.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

1.1 Постановка завдання

Створити веб застосунок для рекомендації закладів харчування у Києві на C#, з використанням фреймворку ML.NET.

В процесі розробки необхідно:

- розібратись з різними типами рекомендаційних систем
- визначити систему для розробки власного застосунку
- створити веб застосунок з використанням ASP.NET
- проаналізувати проблеми та вади мови для розробки рекомендаційної системи
- розібратись та використати фреймворк ML.net для створення процесу рекомендації

РОЗДІЛ 2
ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Рекомендаційні системи

У світі, переповненому інформацією, механізм фільтрації є обов’язковим. Рекомендаційні системи зробили величезний крок до цієї мети, значно покращивши та спростивши життя користувачів. Зі збільшенням кількості онлайн-покупок зросла потреба впевненості в купівлі товарів.

Такі системи корисні як звичайним користувача, так і бізнесу. Системи рекомендацій не тільки покращують взаємодію з користувачами, але й приносять більше доходу для бізнесу.

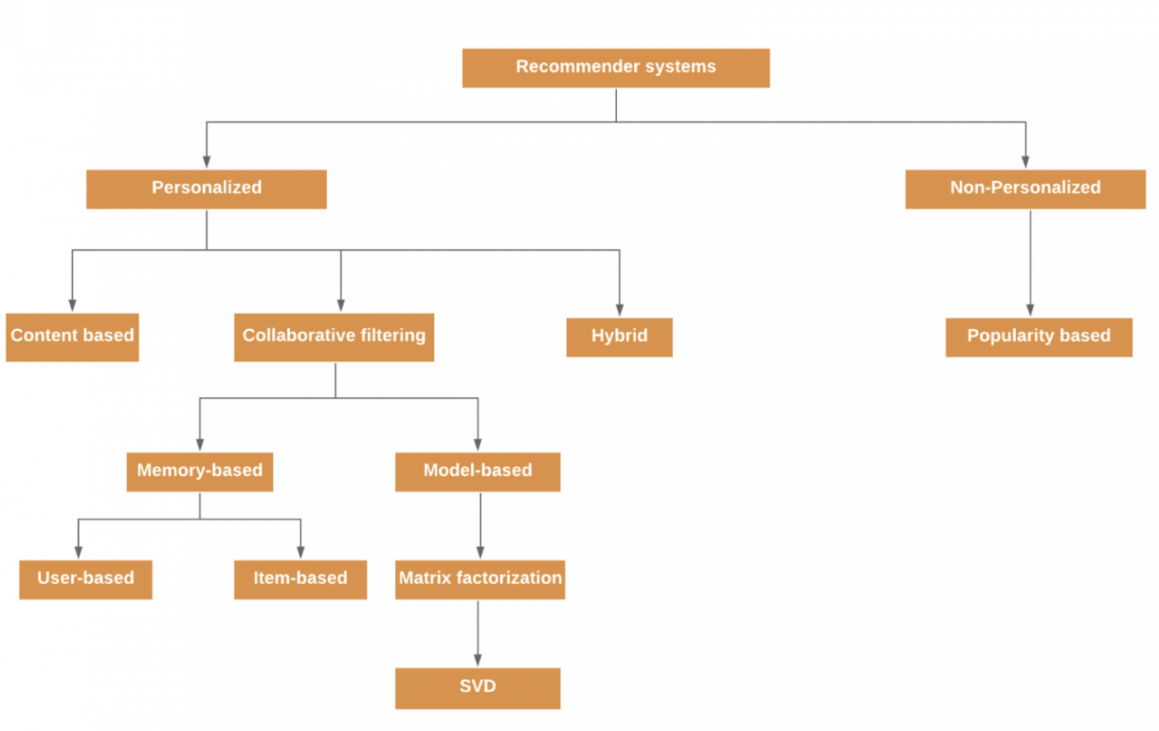
Компанії електронної комерції та роздрібної торгівлі використовують потужність даних для збільшення продажів за допомогою рекомендаційних систем, реалізованих на їхніх веб-сайтах. Саме тому випадки використання цих систем постійно збільшуються.

2.1.1 Типи рекомендаційних систем

Кожна система по своєму унікальна, в тому числі і рекомендаційна. Можемо сказати що рекомендація - це процес, певний алгоритм, який можна змінювати, покращувати та оптимізувати на вимогу бізнесу.
Не існує загального, єдино правильного правила/інструкції, як має бути побудована рекомендаційна система, але розглядаючи загальний принцип

та підхід, можна виділити різні парадигми рекомендаційних систем і розділити їх на типи.

Існує два основних типи рекомендаційних систем – персоналізовані та неперсоналізовані.



В свою чергу серед персоналізованих можна виділити 3 основних принципи, а саме: Content-based (система, що базується на змісті), Collaborative filtering (спільна фільтрація) та Гібридні.

2.1.2 Неперсоналізовані

Неперсоналізовані системи рекомендацій є найпростішими типом рекомендаційних систем. Як підказує сама назва, такі системи не враховують особливості та відмінності користувачів. Рекомендації вироблені цими системами ідентичні для кожного.

Такого типу системи надають користувач рекомендацію на основі того, що інші користувачі/споживачі сказали про предмет/послугу тощо або як оцінили у середньому. Отже це просто пропозиції або список елементів, які користувачеві можуть сподобатися (або ні), і ці рекомендації не залежать від споживач.

Неперсоналізовані рекомендаційні системи в основному використовують два типи алгоритмів: Зведена думка (Aggregated opinion) та рекомендація на основі асоціації продуктів (Basic product association).

2.1.2.1 Зведена думка

Принцип полягає у тому, що ми робимо рекомендацію на основі узагальнення думок натовпу. Тобто використовуємо зведену думку - як рекомендаційний підхід.

Розглянемо для прикладу ресторанний гайд. Кожен ресторан буде мати певний бал (score), що фактично буде дорівнювати середньою з оцінок, присвоєних цьому ресторану іншими клієнтами. Ця оцінка є вимірювання того, наскільки хороший ресторан. Такий підхід дає змогу зрозуміти загальне, однак ці середні показники не мають контексту рекомендації.

Ще одна проблема, що через різну кількість оцінок, у результаті ресторан може отримати хибний бал. Адже при не великій кількості оцінок, одна погана (чи позитивна) оцінка має велику вагу.

2.1.2.2 Асоціативна рекомендація

“Якщо ти вибрав товар_1 (замість товару може бути будь-що), тобі може сподобатись товар_2” - так можна описати, як працює даний тип рекомендації. Неперсоналізовані рекомендації асоціації продуктів можуть надавати корисні неперсоналізовані рекомендації в певному контексті. Більшість інтернет-магазинів, використовують такий підхід. Такі рекомендації засновані на тому, що є в кошику користувача. Ця рекомендація може бути не обов’язково специфічною для користувача, але специфічною для того, що користувач зараз купую (вибирає).

2.1.2.3 Переваги та недоліки неперсоналізованих рекомендацій

Переваги такого підходу в тому, що його легко імплементувати, зібрати та обробити необхідні для цього дані. Однак рекомендації в таких системах однакові для всіх користувачів і не мають персоналізації, а отже, можуть не зацікавити кожного. У випадку, коли ми не маємо жодної інформації про користувача, або коли підхід персоналізованої рекомендації неможливий або не потрібний, такого типу рекомендації є ідеальним рішенням.

2.1.3 Персоналізовані рекомендаційні системи

Персоналізація - це процес створення певного індивідуального підходу, що відповідає клієнтським вподобанням, побудованим на основі поведінки інших користувачів та предметної області.

Така система на відміну від неперсоналізованих аналізує дані користувачів, їхні покупки, рейтинги та шукає спільні риси з іншими користувачами. Таким чином кожен користувач отримає індивідуальні рекомендації.

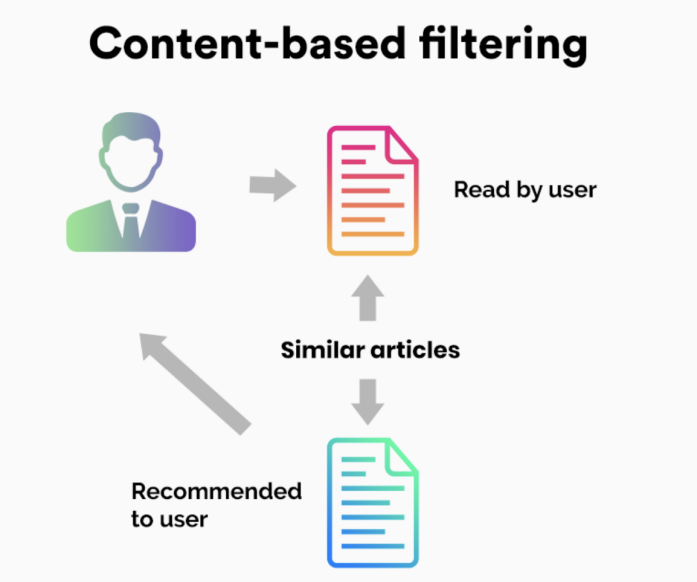
А найпопулярніші типи персоналізованих систем рекомендацій це - фільтрація на основі вмісту та спільна фільтрація.

2.1.3.1 Фільтрація на основі вмісту (Content based)

Фільтрування на основі вмісту використовує функції елементів, щоб рекомендувати інші елементи, схожі на ті, що подобаються користувачеві, на основі їхніх попередніх дій або відгуків (можуть відстежуватись історій покупок користувача).

Методи фільтрації на основі вмісту в основному ґрунтуються на описі елемента та профілю бажаного вибору користувача. У фільтрації на основі вмісту ключові слова використовуються для опису елементів, тоді як профіль користувача створюється для визначення типу елемента, який подобається цьому користувачеві.

Наприклад, якщо користувач уже прочитав книгу від одного автора або купив продукт певної марки, передбачається, що клієнт віддає перевагу цьому автору чи бренду, і існує ймовірність, що користувач придбає подібний продукт у майбутнє.



2.1.3.1.1 Переваги та недоліки фільтрація на основі вмісту

Переваги:

Модель не потребує будь-яких даних про інших користувачів, оскільки рекомендації є специфічними для цього користувача. Це полегшує масштабування для великої кількості користувачів.

Модель може охопити конкретні інтереси користувача та рекомендувати нішеві елементи, які цікавлять не багатьох інших користувачів.

Недоліки:

Оскільки представлення функцій елементів певною мірою розроблено вручну, ця техніка вимагає великих знань предметної області.

Модель може давати лише рекомендації на основі наявних інтересів користувача. Іншими словами, модель має обмежені можливості для розширення наявних інтересів користувачів.

2.1.3.2 Метод/ідея спільної фільтрації (Collaborative filtering)

Метод спільної фільтрації створюється на основі збору та аналізу інформації поведінки, діяльності або вподобань користувачів і передбаченні того, що їм сподобається на основі схожості з іншими користувачами. Прогнозування здійснюється за допомогою різних методів машинного навчання.

Оскільки рекомендація заснована на вподобаннях інших користувачів, вона називається спільною.

Існує два типи спільної фільтрації: на основі пам’яті та на основі моделі.

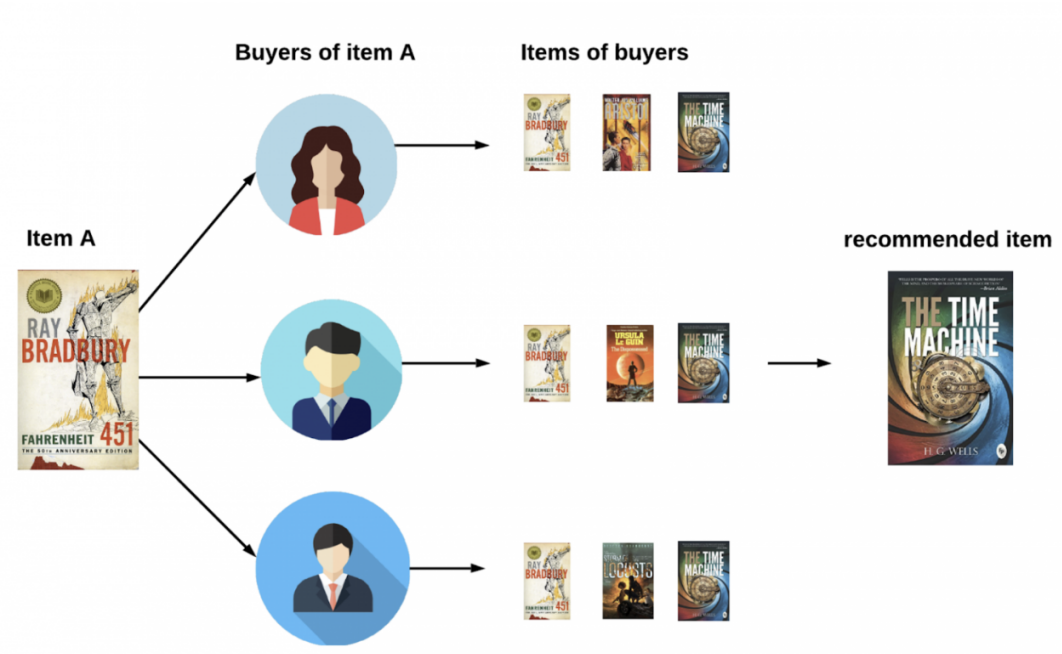
2.1.3.2.1 На основі пам’яті (Memory-based)

Методи, на основі пам’яті, застосовуються до необроблених даних без попередньої обробки і використовують всю базу даних користувачів для створення прогнозу. Вони прості у виконанні, а отримані рекомендації, як правило, легко пояснити. Однак, кожного разу необхідно робити прогнози за всіма даними, що сповільнює систему рекомендації.

Існує два типи: фільтрація на основі користувачів і фільтрація на основі елементів.

2.1.3.2.1.1 Фільтрація на основі елементів (Item-based)

«Користувачам, яким сподобався цей продукт, також може сподобатись…» - так можна описати основний принцип підходу для рекомендації.
Наприклад: Якщо User1, User2 і User3 високо оцінили book1 і book2, наприклад, дали 5 зірок, то коли інший User4 купує book1, тоді book2 також буде йому рекомендована, оскільки система визначила книги як схожі на основі оцінок користувачів.



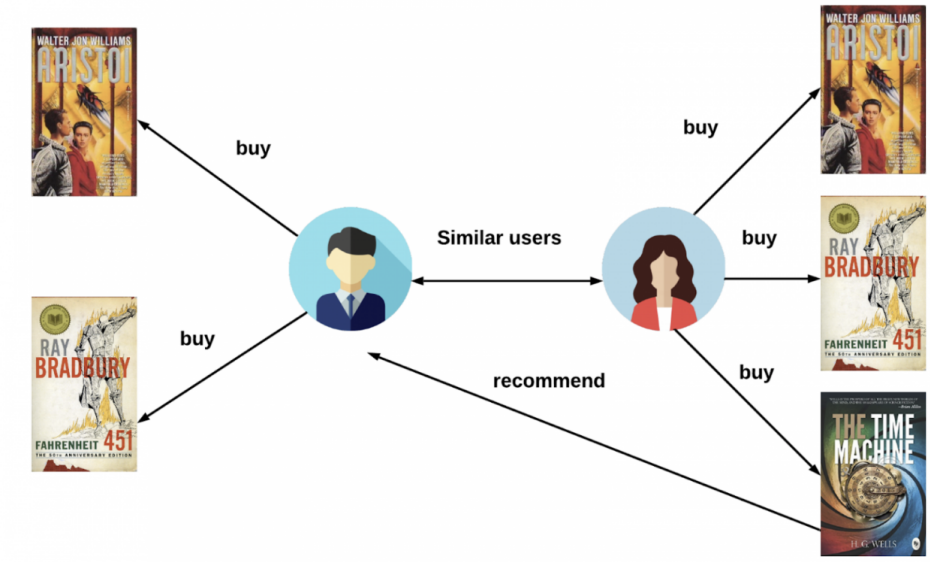
2.1.3.2.1.1 Фільтрація на основі користувачі (User-based)

«Схожим на вас користувачам також сподобались…»

Продукти рекомендовані користувачеві на основі того факту, що вони були придбані/сподобалися користувачам, схожим на користувача якому вони пропонуються. Що означає схожий користувач:
Наприклад, User1 і User2 люблять bookType1 книги. Коли з'являється нова книга bookType1, і User1 купує її, оскільки User2 також любить книги bookType1, ми можемо порекомендувати книгу, яку купив User1.

Недоліком є те, що користувачів, як правило, набагато більше, ніж елементів, що призводить до значно більших матриць подібності користувачів (це може бути зрозуміло в наступному розділі), що призводить до проблем з продуктивністю та пам'яттю для більших наборів даних, що змушує покладатися на методи розпаралелювання або інші підходи взагалі.

Ще одна поширена проблема полягає в тому, що ми страждаємо від холодного запуску : інформації про вподобання нового користувача може бути зовсім небагато, отже, немає з чим порівнювати.



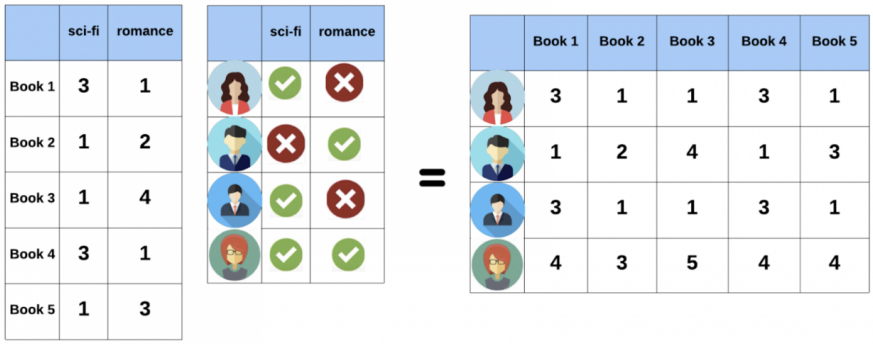
2.1.3.2.1 Рекомендація на основі моделі (Collaborative filtering model based)

Такі моделі були розроблені з використанням алгоритмів машинного навчання. Створюється модель і на основі неї, а не всіх даних, дається рекомендація, що прискорює роботу системи. Такий підхід забезпечує кращу масштабованість. У цьому підході часто використовується зменшення розмірності.

Найвідомішим типом цього підходу є матрична факторизація.

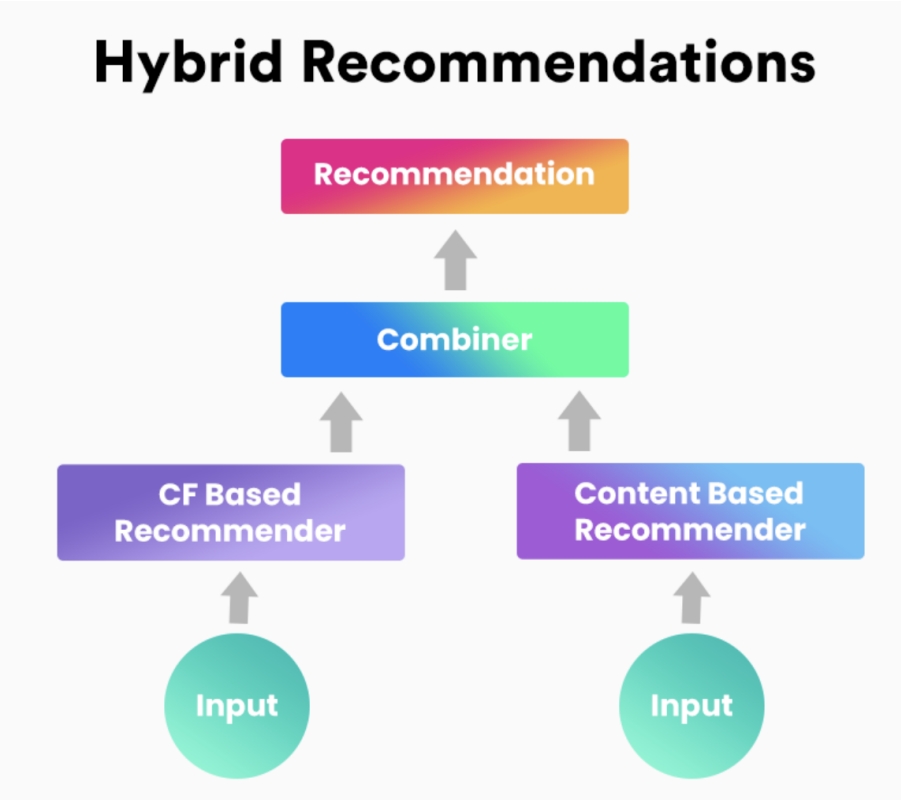
2.1.3.2.2 Матрична факторизація (Matrix factorization)

Якщо, наприклад, є відгук від користувача, користувач переглянув певний фільм або прочитав певну книгу і дав оцінку, яку можна представити у вигляді матриці, де кожен рядок представляє конкретного користувача, а кожен стовпець представляє конкретний предмет. Оскільки практично неможливо, щоб користувач оцінив кожен елемент, ця матриця матиме багато незаповнених значень. Це називається розрідженістю. Методи матричної факторізації використовуються для пошуку набору прихованих факторів і визначення переваг користувача за допомогою цих факторів. Приховану інформацію можна повідомити, аналізуючи поведінку користувачів. Приховані фактори інакше називають ознаками.



2.1.3.3 Hybrid

Механізми гібридних рекомендацій, по суті, є комбінацією різноманітних алгоритмів оцінювання та сортування. Наприклад, гібридна система рекомендацій може використовувати спільну фільтрацію та фільтрацію на основі продуктів у тандемі, щоб рекомендувати клієнтам більш широкий асортимент продуктів з точністю.



2.1.4 Метрики рекомендаційної системи

Який показник буде використаний, залежить від бізнес-задачі, що вирішується. Якщо ми вважаємо, що ми зробили найкращу з можливих рекомендацій, і показник чудовий, але на практиці він поганий, значить, наша система не є хорошою. Найголовніше, щоб користувач отримав довіру до системи рекомендацій. Якщо ми рекомендуємо йому 10 найкращих продуктів, і лише 2 або 3 є для нього актуальними, він вважатиме, що система рекомендацій погана.

Показники:

- Точність (MAE, RMSE)
- Перевірка топ-N рекомендацій:
    - Частота влучень (Hit rate) – спочатку знайдіть усі елементи в історії цього користувача в training data; потім потрібно видалити один із цих пунктів; використовуйте всі інші елементи для рекомендацій і знайдіть 10 найкращих рекомендацій; Якщо вилучений елемент з’являється в 10 найкращих рекомендаціях, то система працює добре.
    - Оцінка частоти влучень - рейтинговий бал для кожного рейтингу розраховується, щоб визначити, який тип рейтингу отримує більше звернень. Підсумуйте кількість звернень для кожного типу рейтингу в топ-N списку та поділіть на загальну кількість елементів кожного рейтингу в списку топ-N.

- A/B Тестування – A/B-тестування є найкращим способом онлайн-оцінки вашої системи рекомендацій.

2.1.5 Проблеми рекомендаційних систем

- Відсутність або недостатня кількість даних

Можливо, найбільша проблема, з якою стикаються системи рекомендацій, полягає в тому, що їм потрібно багато даних, щоб ефективно давати рекомендації. Не випадково найкращими рекомендаціями є компанії з великою кількістю даних користувачів: Google, Amazon, Netflix. Чим більше елементів і даних користувача має працювати з системою рекомендацій, тим більше шансів отримати хороші рекомендації.
Ця проблема особливо актуальна для нових користувачів і відома як холодний старт.

- Зміна смаків користувача та трендів

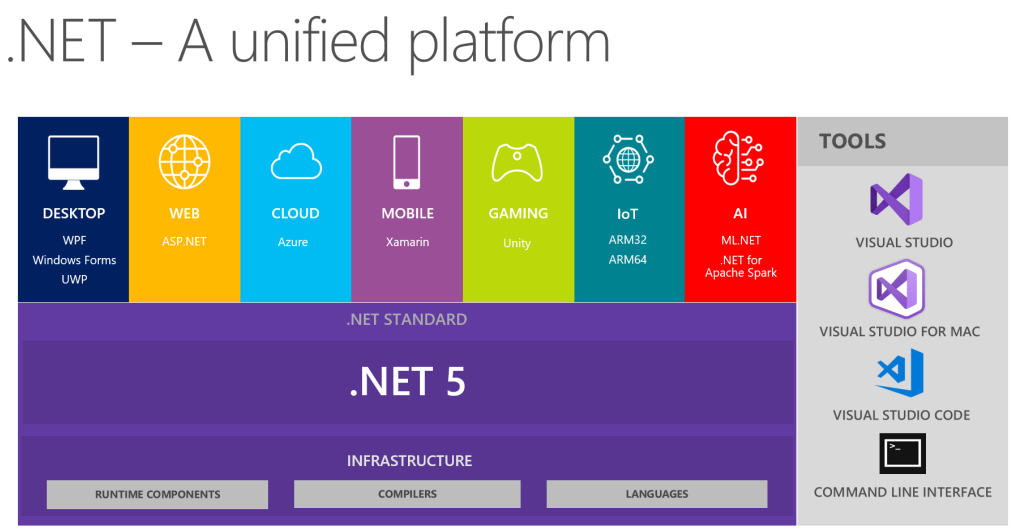
Більшість описаних вище систем працюють таким чином, що намагаються зрозуміти, що користувачеві сподобалось і на основі цього робити припущення, що сподобається в майбутньому і що йому порекомендувати. Однак смаки, так само як і тренди постійно змінюються і товари/послуги, які користувачеві подобались до цього, можуть не подобатись зараз.

- Зміна намірів користувача

Сьогодні користувач може використовувати наш застосунок з одним наміром, а завтра ситуація може кардинально змінитись і вся зібрана на основі попередніх відвідувань інформація буде не корисна для рекомендація.
Наприклад ми завжди користуємось застосунком рекомендації ресторанів, щоб визначити куди сходити. А сьогодні, користувач шукає місце де

відсвяткувати весілля або куди сходити з друзями. Отже з'являються додаткові властивості, які ми не враховуємо.

2.2 Вибір та аналіз мови розробки застосунку



Для розробки застосунку я обрав C#, оскільки це одна з найбільш популярних (5 в рейтингу за 2022 рік), високорівневих мов для розробки застосунків на платформі .Net.

.Net потужна платформа, яка постійно розробляється, дуже швидка, захищена та має велику кількість бібліотек, що полегшують розробку застосунків.

Оскільки мій застосунок, це веб застосунок, я використав ASP.NET фрейвор, а для розробки рекомендаційної частини ML.NET

Оскільки я обрав .NET то вибір БД був очевидним. Я використав SQL Server, а для вищого рівня абстракції взаємодії з БД, я використовував Entity Framework.

2.2.1 ML.NET

ML.NET — це кросплатформна платформа машинного навчання (ML) з відкритим кодом.

ML.NET дозволяє розробникам легко створювати, навчати, розгортати та використовувати власні моделі в своїх програмах .NET, не вимагаючи попереднього досвіду в розробці моделей машинного навчання або досвіду роботи з іншими мовами програмування, такими як Python або R. Фреймворк забезпечує завантаження даних з файлів і баз даних, забезпечує перетворення даних і включає багато алгоритмів ML.

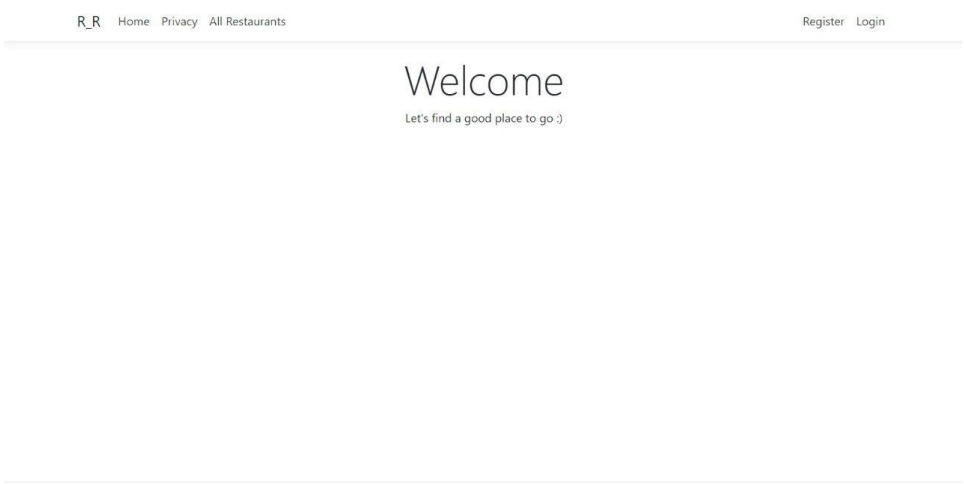
За допомогою ML.NET можна навчати моделі для різних сценаріїв , таких як класифікація, прогнозування та виявлення аномалій.

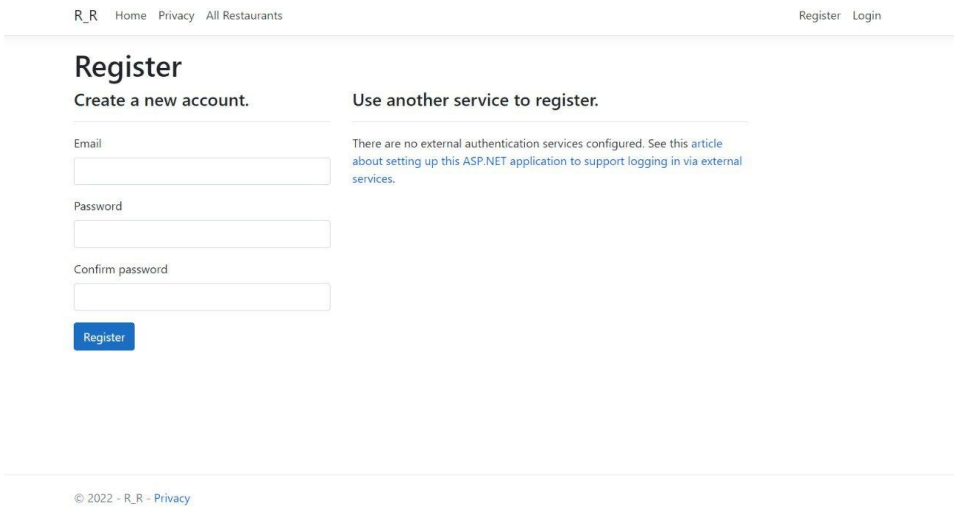
ML.NET був розроблений як розширювана платформа, щоб можна було використовувати інші популярні фреймворки ML (TensorFlow, ONNX, Infer.NET) і мати доступ до ще більшої кількості сценаріїв машинного навчання, таких як класифікація зображень, виявлення об’єктів тощо.

РОЗДІЛ 3
ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Сторінка реєстрації, авторизація та головна сторінка

Перша сторінка на яку потрапляє користувач, це головна сторінка, з якої він має можливість можливість перейти на сторінку входу, для авторизації, зареєструватись або переглянути список всіх ресторанів, як незареєстрований користувач. Незареєстрований користувач має можливість лише переглядати ресторани, та перейти на web сайт закладу.

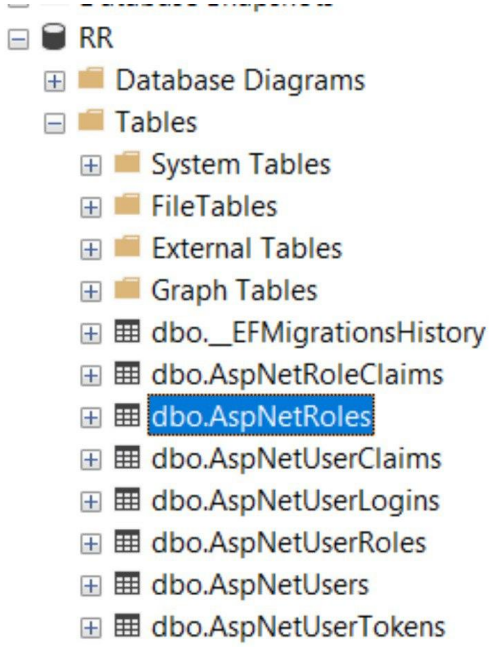




в файлі _Layout.cshtml (що відповідає за шапку сторінки) ми робимо перевірку на те, чи користувач аутентифікований, щоб визначити які компоненти ми будемо показувати. Адже для незареєстрованих користувачі, немає можливості рекомендувати ресторан або планувати майбутній похід.

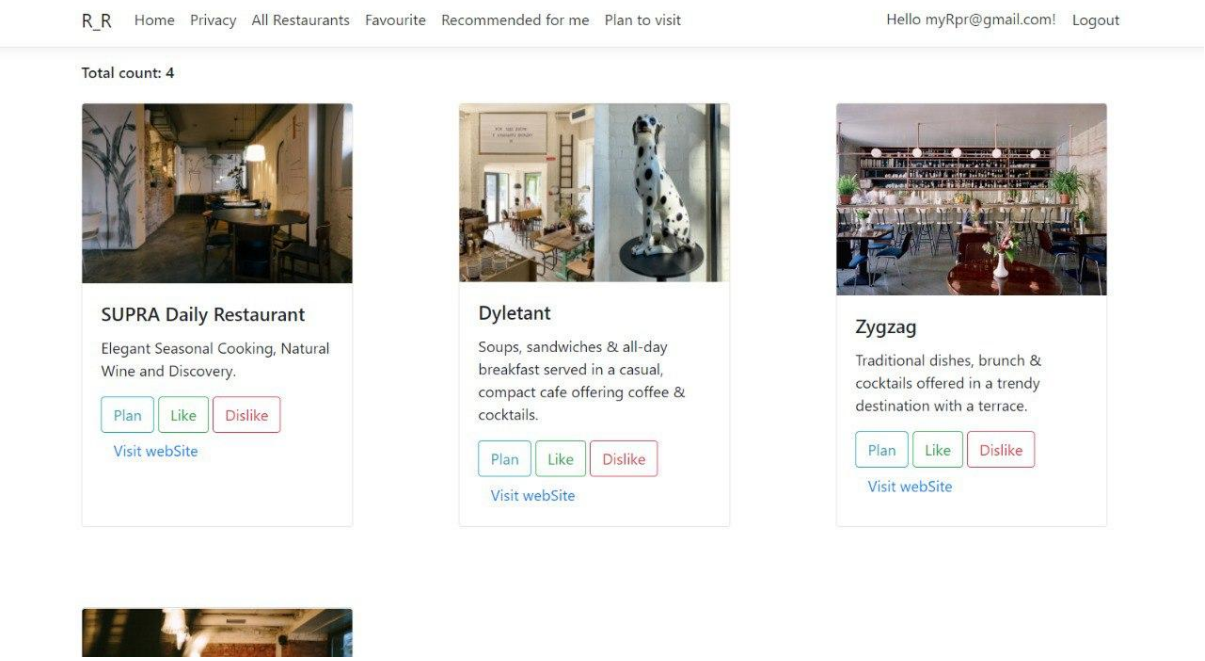
```
            <a class="nav-link text-dark" asp-area="" asp-controller="Restaurant" asp-action="ListOfAll">All Restaurants</a>
        </li>
        @if (User.Identity.IsAuthenticated)
        {
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Restaurant" asp-action="ListOfAll">Favourite</a>
            </li>

            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Restaurant" asp-action="ListOfRecommended">Recommended for me</a>
            </li>

            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Restaurant" asp-action="ListOfAll">Plan to visit</a>
            </li>
        }
    </ul>
    <partial name="_LoginPartial" />
</div>
```

Налаштувавши Connection Properties, при першому запуску програми створюються необхідні для реєстрації та авторизації користувача таблиці в базі нашого SQL Server.



3.2 Сторінка списку ресторанів

Натиснувши на кнопку “All Restaurants” користувач потрапляє на сторінку з усіма ресторанами.




```
0 references
public IActionResult ListOfAll()
{
    ListOfAllViewModel viewModel;
    using (var context = new ApplicationDbContext())
    {

        viewModel = new ListOfAllViewModel { Restaurants = context.Restaurant.ToList() };

    }
    return View(viewModel);
}
```

У користувача є можливість запланувати похід в ресторан/кафе натиснувши кнопку “Plan”, відвідати web сторінку закладу, поставити “Like” або “Dislike”.

```
0 references
public IActionResult Like(string restId)
{
    if (User.Identity.IsAuthenticated == true)
    {
        var currentUserId = User.FindFirstValue(ClaimTypes.NameIdentifier);
        using (var context = new ApplicationDbContext())
        {
            var isExisting = context.UserRestaurantRating.ToList().SingleOrDefault(r => r.UserId == currentUserId &&
                r.RestId == restId && r.Rating == 5) != null;

            if (isExisting == false)
            {
                context.UserRestaurantRating.Add(new Entities.UserRestaurantRating
                {
                    UserId = currentUserId,
                    RestId = restId,
                    Rating = 5,
                    Timestemp = System.DateTime.UtcNow
                });
                context.SaveChanges();
            }
        }

    }

    return Ok();
}
```

Ми робимо перевірку чи користувач аутентифікований, і якщо він поставив лайк, або дизлайк то записуємо у таблицю UserRestaurantRating - id користувача, ресторану та відповідно рейтинг 5 - для лайку та 1 для дізлайку.

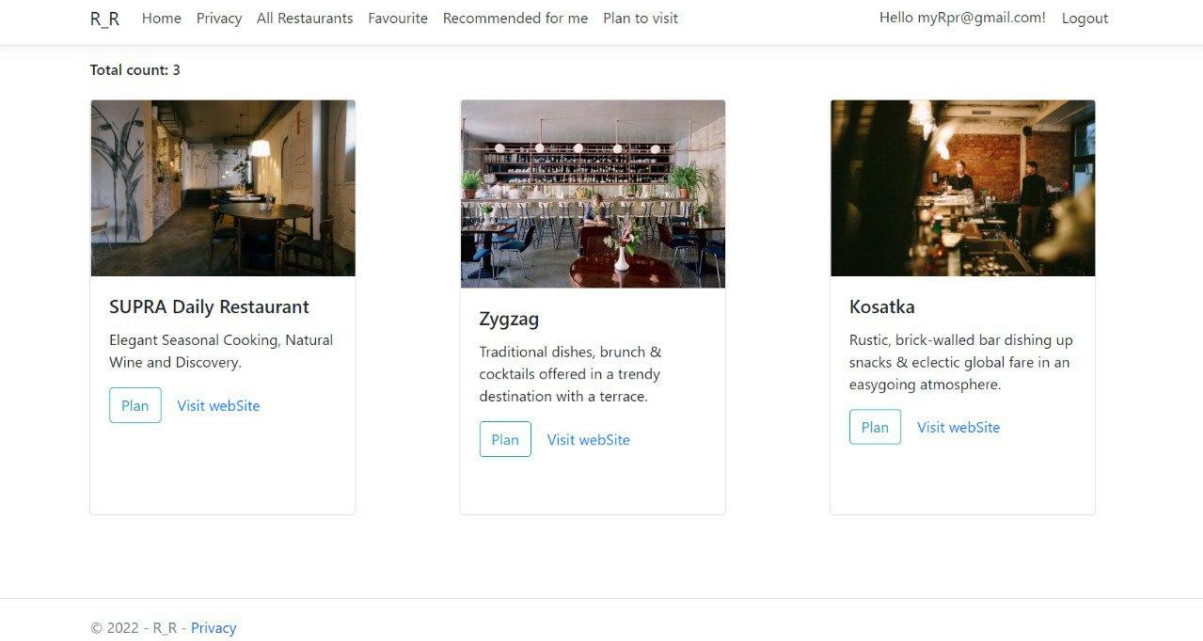
```
SELECT *
FROM [dbo].[UserRestaurantRating]
```

100 %

Results | Messages

	UserId	RestId	Rating	Timestemp	PlanToVisit	Id
1	7fd6841d-7f24-46eb-8893-46d5aef57cc0	102	1	2022-05-18 00:11:26.530	0	2e3f3db3-d535-4b6d-8374-957d9459cb76
2	352c3616-16f4-4463-bbc1-bab25e8615b4	101	5	2022-05-18 00:11:26.530	0	3e57f776-fe77-4bf4-ac63-3265f5dbd57f
3	a36be97f-b934-4233-baf0-6c94c044517d	102	5	2022-05-18 00:11:26.530	0	44278622-8113-4b6b-b508-4fe8c0f23097
4	7fd6841d-7f24-46eb-8893-46d5aef57cc0	101	5	2022-05-18 00:11:26.530	0	4e389788-365f-4637-a7ca-647fccb6cfaa
5	352c3616-16f4-4463-bbc1-bab25e8615b4	103	5	2022-05-18 00:11:26.530	0	533a5956-1be7-4f57-a58c-d028c62eb920
6	a36be97f-b934-4233-baf0-6c94c044517d	104	5	2022-05-18 00:11:26.530	0	5d1e2911-853d-4f8f-81a8-18373990aa94
7	7fd6841d-7f24-46eb-8893-46d5aef57cc0	103	1	2022-05-18 00:11:26.530	0	6e760be5-09ba-43db-9b57-7d0d6982df57
8	d6c9cbde-f30b-4b86-82d9-d074e0001e0a	103	5	2022-05-18 00:11:26.530	0	86468c44-da87-424d-8f27-e4eae9c04582
9	352c3616-16f4-4463-bbc1-bab25e8615b4	102	1	2022-05-18 00:11:26.530	0	86cc08c6-fa80-47a7-9d6b-2aa04d600c1d
10	7fd6841d-7f24-46eb-8893-46d5aef57cc0	104	5	2022-05-18 00:11:26.530	0	8f03c82d-cdd5-4da1-a98e-c55883ffd643
11	a36be97f-b934-4233-baf0-6c94c044517d	102	1	2022-05-18 00:11:26.530	0	a2466cb3-29cb-45f5-b07f-50acea2589c2
12	a36be97f-b934-4233-baf0-6c94c044517d	101	5	2022-05-18 00:11:26.530	0	a5a5fc58-d3a9-4540-9696-449d31d049c9
13	d6c9cbde-f30b-4b86-82d9-d074e0001e0a	101	5	2022-05-18 00:11:26.530	0	a7e93319-a60d-4391-b34f-d315eb76713e

3.3 Рекомендація ресторанів



Після натискання “Recommended for me” відпрацьовує наш алгоритм рекомендації, та повертає список всіх рекомендованих закладів саме для цього, зареєстрованого користувача .

```
0 references
public IActionResult ListOfRecommended()
{
    List<Restaurant> model = new List<Restaurant>();


    if (User.Identity.IsAuthenticated)
    {
        _mLService.Build();
        var userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
        using (var context = new ApplicationDbContext())
        {

            foreach (var rest in context.Restaurant.ToList())
            {
                var isGood=  _mLService.isRestaurantRecommendedForUser(userId, rest.Id);

                if (isGood)
                {
                    model.Add(rest);
                }

            }

        }
    }
    return View(model);
}
```

Перевіривши чи аутентифікований користувач, ми виконуємо метод Build(), що відповідає за підвантаження новий даних будвання та навчання моделі.

```
namespace movierecommender.Services
{
    2 references
    public class MLService : IMLService
    {
        MLContext _mlContext = new MLContext();
        ITransformer _model;
        IConfiguration _conf;

        0 references
        public MLService(IConfiguration conf)
        {
            _conf = conf;
        }

        1 reference
        void SaveModel(DataViewSchema trainingDataViewSchema)
        {
            //Save your trained model by adding the following code in the SaveModel() method:
            var modelPath = Path.Combine(Environment.CurrentDirectory, _conf["MLModelPath"]);
```

Клас MLContext є відправною точкою для всіх операцій ML.NET, а ініціалізація _mlContext створює нове середовище ML.NET, яке можна спільно використовувати між об’єктами робочого процесу створення моделі. Концептуально це схоже DBContext у Entity Framework.

```
(IDataView training, IDataView test) LoadSQLData(MLContext mlContext)
{

    DatabaseLoader loader = mlContext.Data.CreateDatabaseLoader<UserRestaurantRatingML>();

    string connectionString = "data source=SSVAKULENKO-NB; Initial Catalog=RR; Integrated Security=True";

    string sqlCommandTest = "select top (20) percent UserId, RestId, CAST(Rating as REAL) as Rating FROM UserRestaurantRating";
    string sqlCommandTraining = "   select UserId, RestId, CAST(Rating as REAL) as Rating FROM UserRestaurantRating except select top(20) percent " +
        "UserId, RestId, CAST(Rating as REAL) as Rating  FROM UserRestaurantRating";

    DatabaseSource dbSourceTest = new DatabaseSource(SqlClientFactory.Instance, connectionString, sqlCommandTest);
    DatabaseSource dbSourceTraining = new DatabaseSource(SqlClientFactory.Instance, connectionString, sqlCommandTraining);

    IDataView testDataView = loader.Load(dbSourceTest);
    IDataView trainingDataView = loader.Load(dbSourceTraining);


    return (trainingDataView, testDataView);
}
```

Наступним кроком буде завантаження даних з БД. Поділ даних на навчальні та тестові набори є важливою частиною оцінки моделей інтелекту даних. Ми будемо використовувати 20% даних для тестування, а решту 80% для навчання.

Дані в ML.NET представлені у вигляді інтерфейсу IDataView . IDataView - це гнучкий, ефективний спосіб опису табличних даних (числових і текстових).

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


    //recommendation training algorithm
    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = "userIdEncoded",
        MatrixRowIndexColumnName = "restIdEncoded",
        LabelColumnName = "Rating",
        NumberOfIterations = 20,
        ApproximationRank = 100
    };

    var trainerEstimator = estimator.Append(mlContext.Recommendation().Trainers.MatrixFactorization(options));

    /*Console.WriteLine("=============== Training the model ===============");*/
    System.Diagnostics.Debug.WriteLine("=============== Training the model ===============");
    ITransformer model = trainerEstimator.Fit(trainingDataView);

    // save

    SaveModel(trainingDataView.Schema);

    return model;
}
```

Спершу необхідно підготувати та перетворити дані у прийнятний для рекомендаційного алгоритму формат. Оскільки userId та restId представляєте користувачів і назви ресторанів, а не реальні значення, необхідно використати метод MapValueToKey() для перетворення кожного userId та restId в числовий ключ типу Feature і додати їх як нові стовпці набору даних.

userId	restId	Label	userIdEncoded	restIdEncoded
1	2	5	userKey1	restKey1
1	4	5	userKey1	restKey2

Наступним кроком є вибір рекомендаційної системи. Я обрав Matrix Factorization. Підхід до рекомендацій, коли у вас є дані про те, як користувачі оцінювали елементи (ресторани в даному випадку) в минулому.

У цьому випадку Matrix Factorization алгоритм використовує метод, який називається «спільна фільтрація», який передбачає, що якщо користувач 1 має ту ж думку, що й користувач 2, щодо певного закладу, то користувач 1, швидше за все, буде відчувати, мати таку ж симпатію, що й користувач 2.

Matrix Factorization має декілька параметрів, які я використовував:

MatrixColumnIndexColumnName - Ім'я змінної (тобто Column в системі типів IDataView), що використовується як індекс стовпця матриці. Дані стовпця мають бути Single.

MatrixRowIndexColumnName - Ім'я змінної (тобто стовпця в системі типів IDataView), що використовується як індекс рядка матриці. Дані стовпця мають бути KeyDataViewType.

LabelColumnName - Ім'я змінної (тобто стовпець у системі типів IDataView), що використовується як значення елемента матриці. Дані стовпця мають бути KeyDataViewType .

NumberOfIterations - кількість навчальних ітерацій.

ApproximationRank - ранк наближення матриці.

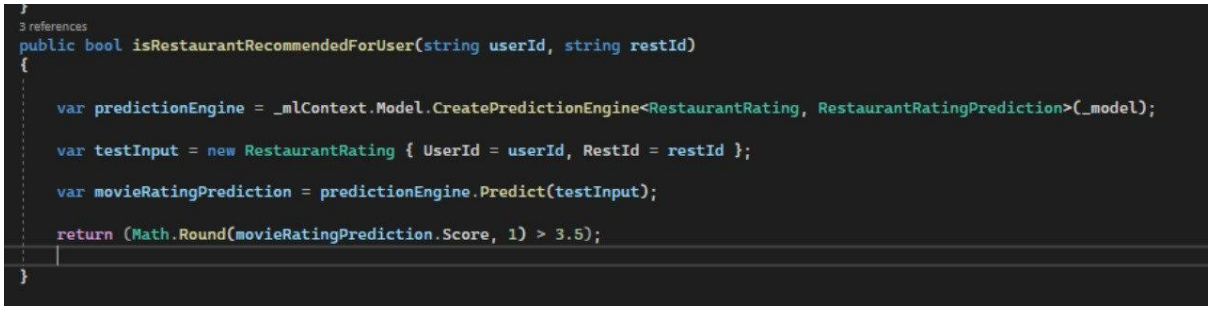
Метод Fit() навчає модель з наданим набором навчальних даних. Він виконує Estimator визначення, перетворюючи дані та застосовуючи навчання, і повертає назад навчену модель, яка є Transformer.

```
void EvaluateModel(MLContext mlContext, IDataView testDataView, ITransformer model)
{
    System.Diagnostics.Debug.WriteLine("=============== Evaluating the model ===============");
    var prediction = model.Transform(testDataView);
    var metrics = mlContext.Regression.Evaluate(prediction, labelColumnName: "Rating", scoreColumnName: "Score");
    System.Diagnostics.Debug.WriteLine("Root Mean Squared Error : " + metrics.RootMeanSquaredError.ToString());
    System.Diagnostics.Debug.WriteLine("RSquared: " + metrics.RSquared.ToString());

}
```

Наступним етапом є оцінка моделі, на основі тестувальних даних, які я вже завантажив, щоб оцінити ефективність своєї моделі.

Коли є набір передбачення, метод Evaluate() оцінює модель, яка порівнює прогнозовані значення з фактичними Labels в наборі даних тесту та повертає показники ефективності моделі.

```
0 references
public IActionResult ListOfRecommended()
{
    List<Restaurant> model = new List<Restaurant>();


    if (User.Identity.IsAuthenticated)
    {
        _mLService.Build();
        var userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
        using (var context = new ApplicationDbContext())
        {

            foreach (var rest in context.Restaurant.ToList())
            {
                var isGood=  _mLService.isRestaurantRecommendedForUser(userId, rest.Id);

                if (isGood)
                {
                    model.Add(rest);
                }

            }

        }
    }
    return View(model);
}
```



```
3 references
public bool isRestaurantRecommendedForUser(string userId, string restId)
{

    var predictionEngine = _mlContext.Model.CreatePredictionEngine<RestaurantRating, RestaurantRatingPrediction>(_model);

    var testInput = new RestaurantRating { UserId = userId, RestId = restId };

    var movieRatingPrediction = predictionEngine.Predict(testInput);

    return (Math.Round(movieRatingPrediction.Score, 1) > 3.5);

}
```

Останнім кроком є саме використання нашої моделі для побудови передбаченя.

Функція Predict() робить прогноз для одного стовпця даних.

Висновки

В процесі розробки курсової роботи я детальніше ознайомився з рекомендаційними системами. Їх різновидами, алгоритмами, підходами реалізації, перевагами та недоліками. Також, я визначив метрики, що допомагають оцінити якісь розробленої рекомендаційної системи та методи їх тестування.

Розробляючи застосунок, я ще раз переконався що C# - ефективна, зручна і універсальна мова. Завдяки великій кількості фреймворків .NET чудово підходить для розробки веб застосунку, навіть з елементами машинного навчання для створення рекомендаційної системи.

Основним випробуванням та званням для мене було обрати правильний алгоритм рекомендації та розібратись з фреймворком ML.NET , що у мене успішно вдалось. Мені сподобалось працювати з цим фреймворком, через хорошу документацію Microsoft та наявність великої кількість методі ML, що дозволило уникнути розробки інфраструктури ML нижнього рівня.

Однак, через те, що цей фреймворк досить новий, крім офіційної документації, в мережі досить важко знайти відповіді на запитання, які у мене виникали під час розробки. Не всі помилки можна було “загуглити”, а прикладів застосунку, певних методів, могло взагалі не бути.
Ще одна проблема, з якою я зіткнувся це специфічне приведення даних до відповідних типів, щоб метод міг використовувати їх для навчання.

Реалізувавши рекомендаційну систему, я зіткнувся ще з однією проблемою, що не стосувалась ML.NET або спицифіки мови. Це відома проблема недоліку даних, відома як холодний старт. Щоб протестувати

алгоритм та переконатись у його ефективності, потрібні великі об'єми даних, яких я не мав.

Отже, рекомендаційні системи - це важливі системи фільтрації даних, що допомагає як бізнесу, та і людям щодня. Її розробка вимагає великої кількості знань в ML, AI, та БД, але результат вартий цих зусиль. А мова C#, завдяки розвитку ML.NET має великий потенціал змагатись в їх розробці навіть з такою мовою як Python.

Джерела

1. https://dotnet.microsoft.com/
2. https://docs.microsoft.com/en-us/dotnet/machine-learning
3. https://towardsdatascience.com/how-does-collaborative-filtering-work-da56ea94e331
4. https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

5. Recommender Systems in e-Commerce: Methodologies and Applications of Data Mining, Dr. Bharat Bhasker , K Srikumar,July 29, 2010.

6. Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl “Analysis of Recommendation Algorithms for ECommerce”, GroupLens Research Group/Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455.

7. Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker and John Riedl, 1999, “Combining collaborative filtering with personal agents for better recommendations,” In Proceedings of the AAAI Conference, pp. 439-446.
8. https://www.sciencedirect.com/science/article/pii/S1110866515000341
9. Mihai Dascalu, (2012) ARSYS-article recommender system