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: Інтелектуальна система аналізу результатів соціологічних досліджень

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

Царик Р.Я.
(прізвище та ініціали)

Керівник *Яцишин С. І.*
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Грицюк Ю.І.
(прізвище та ініціали)

Львів – 2021

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.
"____" _____20___ року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Царик Роман Ярославович
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система аналізу результатів соціологічних досліджень

керівник роботи *Яцишин Світлана Іванівна, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "31" 12.20 року № С-593

2. Термін подання студентом роботи

3. Вихідні дані до роботи

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

Стан проблемної області
Інформаційне та математичне забезпечення
Програмне забезпечення
Технічне забезпечення
Розроблення стартап-проекту

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

6. Дата видачі завдання 18.12.20р, наказ від 31.12.20 N С-593

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
	Стан проблемної області	02.02.21р	**Виконано**
	Інформаційне забезпечення	14.04.21р	**Виконано**
	Математичне забезпечення	11.06.21р	**Виконано**
	Програмне забезпечення	25.07.21р	**Виконано**
	Технічне забезпечення	01.09.21р	**Виконано**
	Розроблення стартап-проекту	28.11.21р	**Виконано**

Студент _____ Царик Р. Я.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яцишин С. І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

В даній дипломній роботі були проаналізовані і оброблені результати соціологічних досліджень. Для обробки даних та прогнозування було використано набір бібліотек, а саме Keras.NET та Torch.NET, котрі дозволяють симулювати роботу нейронної мережі, обробляти та аналізувати дані. Даний програмний продукт був реалізований з використанням актуальних технологій, а саме Angular для візуальної сторони проекту та інтерфейсу користувача, а також .Net 5 – технологія від Microsoft, котра надає чудові можливості для розробки та є однією з найпопулярніших технологій на даний час. Як результат ми отримуємо програмне забезпечення, котре дозволяє обробляти попередньо відформатовані дані та обробляє їх, видаючи користувачеві результати аналізу. Дане програмне забезпечення є цікавим як зі сторони розробки, так і зі сторони застосування, тобто для бізнесу та отримання прибутку. Загалом програмний продукт є вузькоспеціалізованим, але це не мало б йому завадити стати успішним у своїй сфері.

Програмне забезпечення реалізовано за допомогою C#, Angular, HTML 5, SCSS, Bootstrap та набору бібліотек для .NET.

Ключові слова: C#, Angular, інтелектуальна система, аналіз, соціологічні дослідження, Bootstrap.

SUMMARY

In this diploma were analyzed and processed the results of sociological research. A set of libraries, namely Keras.NET and Torch.NET, were used for data processing and forecasting, which allow simulating the operation of the neural network, processing and analyzing data. This software product was implemented using modern technologies, namely Angular for the visual side of the project and user interface, also there was used .Net 5 - a technology from Microsoft that provides excellent development capabilities and is one of the most popular technologies today. As a result, we get software that

allows you to process pre-formatted data and processes it, giving the user the results of the analysis. This software is interesting both in terms of development and application, ie for business and profit. In general, the software product is highly specialized, but this should not prevent it from being successful in its field.

The software is implemented using C #, Angular, HTML 5, SCSS, Bootstrap and a set of libraries for .NET.

Keywords: C #, Angular, intelligent system, analysis, sociological research, Bootstrap.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних
HTML – hypertext markup language
CSS – Cascading Style Sheets
CLI - Command-line interface
UI – user interface
ШІ – штучний інтелект
ШНМ – штучні нейронні мережі

ВСТУП

Актуальність проблеми. Людина – це істота соціальна і в своєму житті сильно залежить від суспільства. Тому соціальні дослідження в значній мірі впливають на суспільство загалом та на окремих індивідів. Оскільки соціальні дослідження мають значний вплив у теперішніх реаліях, то це, безумовно, потрібно аналізувати і певним чином прогнозувати подальші дії опираючись на результати аналізів. Суспільна думка часто може відображати події, котрі відбудуться у подальшому майбутньому, тому такий важливий фактор повинен бути проаналізованим.

Предмет дослідження – технології та інструменти, котрі надають можливість форматувати та аналізувати дані результатів соціологічних досліджень.

Об’єкт дослідження – аналіз результатів соціологічних досліджень.

Мета роботи – розробити інтелектуальну систему аналізу результатів соціологічних досліджень.

Завдання даної роботи полягає у тому, щоб розробити систему, котра дозволить, попередньо обробивши дані соціологічних досліджень, проаналізувати ці дані видати певний результат на основі проробленої роботи.

Наукова новизна – полягає в тому, що створене програмне забезпечення дозволяє працювати з будь-якими результатами соціологічних досліджень, попереднього обробивши їх і привівши до сумісного формату.

Практичне значення результатів полягає у тому, що на ринку немає безоплатних аналогів даного програмного забезпечення.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Соціологічні дослідження

Соціологія вивчає суспільне життя, соціальні зміни, соціальні причини та наслідки поведінки людей. Соціологи досліджують структуру груп, організацій і суспільств, а також те, як люди взаємодіють у цих контекстах. Оскільки вся людська поведінка є соціальною, предмет соціології коливається від близької для сім'ї до ворожої для натовпу; від організованої злочинності до релігійних культів; від поділу раси, статі та соціального класу до спільних вірувань та спільної культури; і від соціології праці до соціології спорту. Насправді, небагато галузей мають такі широкі масштаби та актуальність для досліджень, теорії та застосування знань.

Соціологія надає багато різних точок зору на світ, генеруючи нові ідеї та критикуючи старі. Сфера також пропонує цілий ряд дослідницьких методів, які можна застосувати практично до будь-якого аспекту соціального життя: вулична злочинність і правопорушення, скорочення штату компаній, як люди виражають емоції, реформа добробуту чи освіти, як сім'ї відрізняються та процвітають, або проблеми миру та війни. Оскільки соціологія вирішує найскладніші проблеми сучасності, це сфера, яка швидко розширюється, потенціал якої все більше використовується тими, хто розробляє політику та створює програми. Соціологи розуміють соціальну нерівність, моделі поведінки, сили соціальних змін і опору, а також те, як працюють соціальні системи. Як свідчать наступні сторінки, соціологія – це захоплююча дисципліна з розширюючими можливостями для широкого кола кар'єрних шляхів.[1].

Висновки до розділу

Вивчення соціології дозволить вам дізнатися про широкий спектр тем, які часто є основними проблемами в українському суспільстві та й в усьому світі.

Найбільше значення соціології полягає в тому, що вона науково вивчає соціальні інститути суспільства. Останнім часом усвідомлюється значення соціології як науки про людські стосунки. Науковим вивченням суспільства та

науковим сприянням добробуту людей тривалий час нехтували. Тепер же наукове вивчення суспільства йде повним ходом. Це століття має бути століттям розвитку людського та соціального добробуту, якщо ми хочемо досягти соціального прогресу. Тому багато хто справедливо вважає, що соціологія може бути найкращим підходом до всіх соціальних наук і, отже, ключовим дослідженням для нинішньої ситуації.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Результати соціологічних опитувань

Суспільство перебуває на важливому перетині у боротьбі з викликами зміни клімату, і ця стаття представлена на критичному етапі у світлі зростаючого визнання того, що природничих наук недостатньо для вирішення цих проблем. Критичні аспекти соціологічних перспектив, пов'язаних із дослідженнями зміни клімату, об'єднані в цьому огляді з надією сприяти більшій міждисциплінарній співпраці між природними та соціальними науками. Ми палко стверджуємо необхідність впровадження міждисциплінарних підходів, які можуть надати інноваційні перспективи та рішення проблем, з якими ми стикаємося внаслідок впливу зміни клімату. Таким чином, розглядаються деякі критичні соціологічні перспективи з двома цілями: (a) надати основне відкриття для читачів, які шукають вступну перспективу та потенційний основний внесок соціологічного розуміння зміни клімату; і (b) досліджувати можливості та перешкоди, які можуть виникнути при посиленні міждисциплінарної співпраці та співпраці. Ми викладаємо фундаментальні ідеї, збираючи слабко пов'язаний корпус соціологічних досліджень, сподіваючись розробити та просувати програму спільних досліджень між соціологією та іншими дисциплінами на найближче майбутнє.

Зміна клімату є досить важливою проблемою. Через це хвилюються як фахівці, так і звичайне населення. Близько 84% українців впевнені, що зараз на нашій планеті відбуваються глобальні зміни клімату. Близько половини учасників опитування відповіли, що для них це питання є вкрай важливим. То ж можна сказати, що суспільство помічає і розуміє, що відбуваються зміни і вважає ці зміни критичними. Дані були оприлюднені Соціологічною службою компанії R@B Group.

Якщо опиратися на дані цього соцопитування, то можна побачити, що не згоден з цими твердженнями кожен десятий житель України і це становить приблизно – 11%, а також ми бачимо, що ще 5% опитаних не визначились.

Ще одним з результатів є виявлення того факту, що в меншій мірі глобальні проблеми зі змінами клімату визнають особи чоловічої статі і жителі західного частини країни, а частіше – особи жіночої статі, жителі України у віковій категорії 40–50 років, особи котрі проживають у містах (не враховуючи обласних центрів), а також жителі центральної і південної частин нашої країни.

Також потрібно зауважити, що більша частина жителів України, а саме 78% є стурбовані глобальними зміна клімату та наслідків цього процесу. Але з іншого боку це не викликає жодного хвилювання в кожного п’ятого українця, а саме 18%. Решта 4% ще не визначились з позицією. Варто зауважити, що у порівнянні з 2012 роком українці турбуються значно більше про зміну кліматичних умов.

Опитування було проведене з 11 до 20 липня 2020 року. Це опитування проводилося методом особистого інтерв’ю і в результаті було опитано майже дві тисячі респондентів, що старші 18 років на території майже усієї України[5].

2.2 C# (Сі-Шарп)

C# — це об’єктно-орієнтована мова програмування від Microsoft, яка має на меті поєднати обчислювальну потужність C++ із легкістю програмування Visual Basic. C# заснований на C++ і містить функції, подібні до Java.

C# розроблено для роботи з платформою Microsoft .NET. Метою Microsoft є полегшення обміну інформацією та послугами через Інтернет, а також надання розробникам можливості створювати дуже портативні програми. C# спрощує програмування завдяки використанню Extensible Markup Language (XML) і Simple Object Access Protocol (SOAP), які дозволяють отримати доступ до об'єкта або методу програмування, не вимагаючи від програміста писати додатковий код для кожного кроку. Оскільки програмісти можуть використовувати наявний код, а не повторювати його, очікується, що C# зробить його швидшим і менш дорогим для виведення нових продуктів і послуг на ринок.

Microsoft співпрацює з ECMA, міжнародним органом стандартів, щоб створити стандарт для C#. Визнання C# Міжнародною організацією зі стандартизації (ISO) спонукало б інші компанії розробляти власні версії мови. Компанії, які вже використовують C#, включають Apex Software, Bunka Orient, Component Source, devSoft, FarPoint Technologies, LEAD Technologies, ProtoView та Seagate Software. [6].

Найголовніше, C# розроблено спеціально для Microsoft .NET Framework. Це дозволяє розробникам скористатися всіма функціями, які пропонує .NET API. Однак це також означає, що програми C# можуть працювати лише на платформах, які підтримують середовище виконання .NET, таких як Windows, Windows Server і Windows Phone. Щоб програми, написані на C#, працювали на інших платформах, код повинен бути скомпільований за допомогою інструмента перетворення, наприклад Microsoft .NET Native. [7].

2.3 Платформа .NET Core

З останнім запуском ASP.NET Core Microsoft також пішла шляхом Windows, щоб надати користувачам можливість розробки веб-додатків на інших платформах. Високопродуктивні, кросплатформні функції фреймворка з відкритим вихідним кодом, передова функціональність дозволяють створювати хмарні веб-додатки.

Раніше ASP.NET 4.x мав величезний успіх у розробників по всьому світу. Але ASP.NET Core — це оновлена версія з більш модульною та більш компактною архітектурною структурою. Однак ви можете побачити багато дивовижних функцій в останній версії .NET 6.

Відкриваючи шлях для ефективної та простої розробки веб-додатків, ASP.NET Core — це потужний комплекс розширених функцій, який здобув величезну популярність серед розробників. Високопродуктивний кросплатформний фреймворк сьогодні широко використовується для створення хмарних і сучасних додатків.

Використовуючи це передове програмне забезпечення, ви можете:

- Запуск веб-додатків на .NET Framework або .NET Core.
- Розробляйте прогресивні та продуктивні веб-додатки та послуги, мобільні серверні системи та додатки Інтернету речей.
- Підтримка кількох платформ, оскільки надає можливість створювати програми для Windows, Linux та macOS.
- Отримайте гнучкість для розгортання програм і служб локально або в хмарі.

Коли справа доходить до .NET Core, це найкраща платформа для веб-розробки. З розробкою ASP.NET Core, Microsoft вирішила багато питань одним пострілом. Це дозволяє розробникам розробляти веб-програми, веб-сервіси, мобільний сервер та багато інших речей в рамках єдиної платформи. [2].

2.4 TypeScript

TypeScript використовується в Angular (одному з найбільших інтерфейсних фреймворків). Історично JavaScript став основною мовою для написання сценаріїв веб-сторінок і програм в Інтернеті. Тепер можна використовувати JavaScript як на інтерфейсі, так і на серверній частині за допомогою таких фреймворків, як Node.js і Deno.

Але JavaScript не був створений для написання великих складних систем, подібних до сучасної мережі. TypeScript був випущений як відкритий код у 2012 році після розробки в Microsoft. (Програмний гігант залишається розпорядником проекту та головним розробником.) Ця стаття ZDNet того часу пропонує інтригуючий погляд на те, чому це сталося: «Виявляється, однією з головних мотивацій був досвід інших команд Microsoft, які намагалися розробити і підтримувати продукти Microsoft на JavaScript».

Коротше кажучи, TypeScript використовується, коли вам потрібні корпоративні функції та інструменти такої мови, як Java, але вам потрібен ваш код для виконання в середовищі JavaScript. Теоретично ви можете написати стандартний JavaScript, який компілятор TypeScript генерує самостійно, але це

займе набагато більше часу, а великій команді буде складніше спільно зрозуміти та налагодити кодову базу.

TypeScript має ще один чудовий трюк в рукаві: ви можете налаштувати компілятор на конкретне середовище виконання JavaScript, браузер або навіть мовну версію. Оскільки будь-який добре сформований код JavaScript також є кодом TypeScript, ви можете, наприклад, взяти код, написаний у специфікації ECMAScript 2015, яка включала низку нових синтаксичних функцій, і скомпілювати його в код JavaScript, який буде сумісний із застарілими версіями мову. [3].

2.5 Angular

Angular — це фреймворк JavaScript, створений для створення інтерфейсів на стороні клієнта, і він також інструктує розробника про те, як має бути створено веб-додаток. Іншими словами, цей фреймфорк є упевненим. У нього своя філософія і набір правил, які розробник повинен зрозуміти, перш ніж працювати з ним.

Побудований за аналогічними принципами, Angular також створений для роботи як фреймворк із концепцією Model-View-Controller. Початковий запуск Angular був спрямований на полегшення динамічного оновлення сторінок, що відокремлює логіку програми від маніпуляції DOM.

Протягом перших років Angular допомагав розробникам створювати високопродуктивні та чисті односторінкові програми з багатими функціями та функціоналом. Не кажучи вже про Angular, що вніс зручність у всю систему.

Крім того, ви повинні знати, що Angular побудований на TypeScript, а також має екосистему інтегрованих бібліотек, які призначені для охоплення широкого спектру функцій. Серед них, зокрема, маршрутизація, керування формами, зв’язок між клієнтом і сервером.

Нарешті, Angular має вбудований набір інструментів, орієнтованих на розробників, які дозволяють розробляти, тестувати та масштабувати додаток, коли це потрібно. На сьогоднішній день одним із найнеймовірніших прикладів Angular є Gmail. Так, Gmail створено за допомогою Angular.

Angular полегшує створення односторінкових програм і масштабування їх до програм корпоративного рівня. Використання Angular означає легкі оновлення, що корисно, коли нові розробки додаються до джерела, і те ж саме можна легко відобразити в програмі. Однією з найкращих переваг роботи з Angular є те, що ви отримаєте доступ до спільноти з 1,7 мільйона розробників Angular.

Спільнота включає не лише розробників, а й творців контенту, коучів, авторів бібліотек та кількох інших професіоналів, які допомагають кожному навчатися, розвиватися та розвиватися в Angular. [4].

2.6 Нейронні мережі

Нейронні мережі відображають поведінку людського мозку, дозволяючи комп'ютерним програмам розпізнавати закономірності та вирішувати загальні проблеми в області ШІ, машинного навчання та глибинного навчання.

Нейронні мережі, також відомі як штучні нейронні мережі або змодельовані нейронні мережі, є підмножиною машинного навчання і лежать в основі алгоритмів глибинного навчання. Їх назва та структура натхненні людським мозком, імітуючи спосіб, яким біологічні нейрони сигналізують один одному[20].

Штучні нейронні мережі (ШНМ) складаються з шарів вузлів, що містять вхідний шар, один або кілька прихованих шарів і вихідний шар. Кожен вузол або штучний нейрон з'єднується з іншим і має відповідну вагу та поріг. [8].

Нейронні мережі покладаються на навчальні дані, щоб з часом вивчати та покращувати свою точність. Однак, як тільки ці алгоритми навчання будуть налаштовані на точність, вони стануть потужними інструментами в інформатиці та ШІ, що дозволить нам класифікувати та кластерувати дані з високою швидкістю. Завдання розпізнавання мовлення або зображення можуть займати хвилини та години, якщо порівнювати з ручною ідентифікацією експертів. Однією з найвідоміших нейронних мереж є пошуковий алгоритм Google[9].

2.7 Машинне навчання

Машинне навчання змушує комп’ютери самостійно програмувати. Якщо програмування — це автоматизація, то машинне навчання — це автоматизація процесу автоматизації. Написання програмного забезпечення – це вузьке місце, де у нас не вистачає хороших розробників. Нехай дані виконують роботу замість людей. Машинне навчання – це спосіб зробити програмування масштабованим.

Машинне навчання схоже на фермерство чи садівництво. Насіння — це алгоритми, поживні речовини — це дані, садівник — це ви, а рослини — це програми.

Кожен алгоритм машинного навчання складається з трьох компонентів:

Представлення: як представляти знання. Приклади включають дерева рішень, набори правил, екземпляри, графічні моделі, нейронні мережі, машини опорних векторів, ансамблі моделей та інші.

Оцінювання: спосіб оцінки програм-кандидатів (гіпотез). Приклади включають точність, передбачення та відкликання, квадрат помилки, ймовірність, апостеріорну ймовірність, вартість, маржу та інші.

Оптимізація: спосіб створення програм-кандидатів, відомий як процес пошуку. Наприклад, комбінаторна оптимізація, опукла оптимізація, оптимізація з обмеженнями.

Висновки до розділу

Можна сміливо опиратися на дані надані центром беручи до уваги стратегічні цілі центрів котрі проводять і надають дані для інформаційного забезпечення:

- Підтримка правдивої думки суспільства в громадському просторі;
- Збір інформації, опитування та обробка зібраної інформації;
- Інформування громадськості стосовного позиції більшості;
- Формування незалежних і правдивих опитувань, котрі не несуть жодного підтексту чи прихованих мотивів.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Основні концепції машинного навчання

3.1.1 Інженерія даних

Інженерія даних є одним із аспектів науки про дані, який в основному зосереджений на додатках для збору даних, зборі даних та аналізі даних. Вся робота, яку виконують науковці з даних, які люблять відповідати на кілька питань, пов’язаних із прогнозами чи аналізом, використовує великий набір інформації[18].

Тепер їм потрібна правильна та корисна інформація, що створює потребу в зборі та перевірці наявної інформації. Все це є частиною інженерних завдань. Деякі з цих завдань – це перевірка нульових значень (відсутні дані), категоризація даних (категорійні дані), створення структур даних (правила асоціації) тощо[10].

3.1.2 Візуалізація даних

Візуалізація даних – це графічний підхід до представлення даних. Тут ми використовуємо бібліотеку Angular для створення візуальних елементів, наприклад, таблиць, кореляційних діаграм, стовпчастих графіків, парних графіків тощо; Візуалізація даних відіграє важливу роль у забезпеченні простого способу аналізу даних, перегляду та розуміння тенденцій, тощо[11].

3.1.3 Статистичне розуміння

Статистика відіграє важливу роль у сфері науки про дані. Статистика є потужним інструментом для виконання завдань науки про дані. Статистика використовує математику для технічного аналізу доступної інформації. За допомогою візуалізацій, таких як стовпчик або діаграма, ми можемо отримати інформацію про тенденції, але статистика допомагає нам оперувати даними математичним/цільовим способом. Без знання даних наукова візуалізація є просто грою у вгадування[12].

3.2 Методи машинного навчання

3.2.1 Регресія

Алгоритми регресії здебільшого використовуються для прогнозування певних чисел, тобто коли результат є дійсним або безперервним значенням. Оскільки він підпадає під контроль за навчанням, він працює з навченими даними для прогнозування нових даних тесту. Наприклад, вік може бути безперервним значенням, оскільки він збільшується з часом[13].

Так алгоритми часто використовують в методах регресії:

- **Проста модель лінійної регресії**: це статистичний метод, який аналізує зв’язок між двома кількісними змінними. Регресійний аналіз використовує математичні моделі для опису відносин.
- **Логістична регресія**: метод статистичного аналізу, який використовується для прогнозування значення даних на основі попередніх спостережень за набором даних. Модель логістичної регресії передбачає залежну змінну даних шляхом аналізу зв'язку між однією або кількома існуючими незалежними змінними.
- **Регресія Ласо**: Оператор згортання найменшого абсолютного вибору або LASSO використовується, коли існує потреба в підмножині предиктора для мінімізації помилки передбачення в безперервній змінній.
- **Регресія вектора підтримки**: У простій регресії мета полягає в тому, щоб мінімізувати помилку, тоді як у регресії вектора підтримки ми коригуємо помилку в межах порогу.
- **Алгоритм множинної регресії**: цей алгоритм призначений для роботи з кількома кількісними змінними як у різних алгоритмах регресії (лінійних та нелінійних).
- **Алгоритм багатовимірної регресії**: цей метод використовується у випадку кількох змінних-провісників. Ним можна керувати матричними операціями та різними бібліотеками у певних мовах програмування.

З усіх цих алгоритмів регресії у нашому програмному забезпеченні використовується лише алгоритм багатовимірної регресії. Багатовимірна регресія

є одним із найпростіших алгоритмів машинного навчання. Він відноситься до класу алгоритмів контрольованого навчання, тобто коли нам надається набір навчальних даних[19].

Використання можна поділити на кілька кроків:

- Вибір ознак. Знаходження ознак, від яких залежить (чи ні) змінна відповіді. Це є одним із найважливіших кроків багатовимірної регресії. Щоб спростити наш аналіз, ми припустимо, що ознаки, від яких залежить змінна відповіді, вже вибрані.

- Нормалізація ознак: об'єкти потім масштабуються, щоб привести їх у діапазон (0,1) для кращого аналізу. Відобразити це можна за допомогою функції:

$$X_i = \frac{x_i - \mu_i}{\delta_i} \quad (1)$$

де x_i – це навчальний приклад для його особливості, μ_i - означає його особливість, δ_i - діапазон його особливостей.

- Вибір функції гіпотези та вартості. Гіпотеза — це передбачене значення змінної відповіді, представленої як h(x). Функція вартості визначає вартість помилкового прогнозування гіпотези. Він повинен бути якомога меншим. Вибираємо функцію гіпотези як лінійну комбінацію ознак X.

$$h\left(x^i\right) = \theta_0 + \theta_1 x_1^i + \cdots + \theta_n x_n^i \quad (2)$$

де $\theta = [\theta_0 + \theta_1 + \cdots + \theta_n]^T$ це параметр вектора, а x_n^i – це особливість його навчального прикладу. І функція вартості як сума квадратів помилки за всіма навчальними прикладами:

$$J(\theta) = \frac{1}{2m * \sum(h_\theta(x^i) - y^i)^2} \quad (3)$$

- Мінімізація функції витрат: далі деякий алгоритм мінімізації витрат працює над наборами даних, який коригує параметри гіпотези. Після того, як функція вартості зведена до мінімуму для навчального набору даних, її також слід мінімізувати для довільного набору даних, якщо відношення є універсальним. Алгоритм градієнтного спуску є хорошим вибором для мінімізації функції вартості у випадку багатовимірної регресії[21].

3.2.2 Кластеризація

Кластеризація — це завдання поділу множини або точок даних на кілька груп таким чином, щоб точки даних в тих самих групах були більш схожими на інші точки даних у тій же групі, ніж в інших групах. Простіше кажучи, мета полягає в тому, щоб відокремити групи зі схожими ознаками та об’єднати їх у кластери.

Кластеризація відноситься до алгоритмів для виявлення таких кластерів у немаркованих даних. Точки даних, що належать до одного кластеру, мають схожі характеристики, тоді як точки даних з різних кластерів відрізняються один від одного. Ідентифікація таких кластерів призводить до сегментації точок даних на ряд окремих груп. Оскільки групи ідентифікуються з самих даних, на відміну від відомих цільових класів, кластеризація розглядається як навчання без нагляду.

Кластеризація методом к-середніх є, мабуть, найпопулярнішим алгоритмом кластеризації. Це метод розподілу, який поділяє простір даних на K окремих кластерів. Він починається з випадково вибраних K центрів, а всі точки даних призначаються найближчим центрам кластерів. Потім центри кластерів перераховуються як центроїди новоутворених кластерів. Точки даних повторно призначаються до найближчих центрів кластерів, які ми щойно перерахували. Цей процес, призначаючи точки даних центрам кластерів і перераховуючи центри кластерів, повторюється до тих пір, поки центри кластерів не перестануть рухатися.

Цей алгоритм, якщо коротко, працює так:

1. Вкажіть кількість потрібних кластерів.

2. Потрібно вказати початкова значення центроїдів, перемішавши даних і згодом випадковим чином вибравши K (кількість) точок даних для центроїдів не використовуючи заміни.

3. Потрібно робити те ж саме доки центроїди не будуть змінені (призначення точок даних кластерам не змінюється).

4. Вирахуйте суму квадратів дистанції між точками даних і всіма іншими центроїдами.

5. Далі потрібно призначити кожну з існуючих точок даних кластеру (центроїду), котрий знаходиться найближче.

6. Наступним кроком потрібно обрахувати центроїди для кластерів. Це робиться через середнє значення всіх точок даних, які належать кожному кластеру.

Метод описується ось такою функцією:

$$J = \sum_{i=1}^{m}\sum_{k=1}^{K} \omega_{ik}||x^i - \mu_k||^2 \quad (4)$$

де ω_{ik}=1 для точки даних x^i, якщо вона належить кластеру k; інакше ω_{ik}=0. Крім того, μ_k— центроїд кластера x^i.

Оскільки алгоритми кластеризації, використовують вимірювання на основі відстані для визначення подібності між точками даних, рекомендується стандартизувати дані, щоб вони мали середнє значення нуль і стандартне відхилення одиниці, оскільки майже завжди ознаки в будь-якому наборі даних будуть мати різні одиниці виміру. наприклад, вік проти доходу[14].

Враховуючи ітераційний характер методу і випадкову ініціалізацію центроїдів на початку алгоритму, різні ініціалізації можуть призвести до різних кластерів, оскільки алгоритм може застрягти в локальному оптимумі і не сходитися до глобального оптимуму. Тому рекомендується запустити алгоритм, використовуючи різні ініціалізації центроїдів і вибрати результати запуску, які дали меншу суму квадратів відстані[15].

Призначення прикладів не змінюється – це те саме, що і відсутність змін у варіації всередині кластера:

$$\frac{1}{m_k}\sum_{i=1}^{m_k} ||x^i - m_c k||^2 \quad (5)$$

Висновок до розділу

Машинне навчання є підсферою штучного інтелекту. Замість того, щоб покладатися на явне програмування, це система, за допомогою якої комп’ютери

використовують величезний набір даних і застосовують алгоритми, щоб «тренуватися» — навчатися — і робити прогнози.

Бізнес-додатків машинного навчання багато, але всі вони зводяться до одного типу використання: обробка, сортування та пошук шаблонів у величезних обсягах даних, які людям було б непрактично зрозуміти.

Системи машинного навчання здатні швидко застосовувати знання та навчання з великих наборів даних, щоб досягти успіху в розпізнаванні обличчя, мови, розпізнаванні об’єктів, перекладі та багатьох інших завданнях.

Мета машинного навчання, тісно поєднана з метою штучного інтелекту. Вона полягає в досягненні повного розуміння природи процесу навчання (як навчання людини, так і інших форм навчання), обчислювальних аспектів поведінки при навчанні та впровадження можливостей навчання в комп'ютерних системах.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Налаштування .NET Core проекту

Ми створимо проект .Net Core за допомогою середовища розробки Visual Studio. Потрібно встановити необхідні інструменти (Рисунок4.1 та Рисунок4.2).

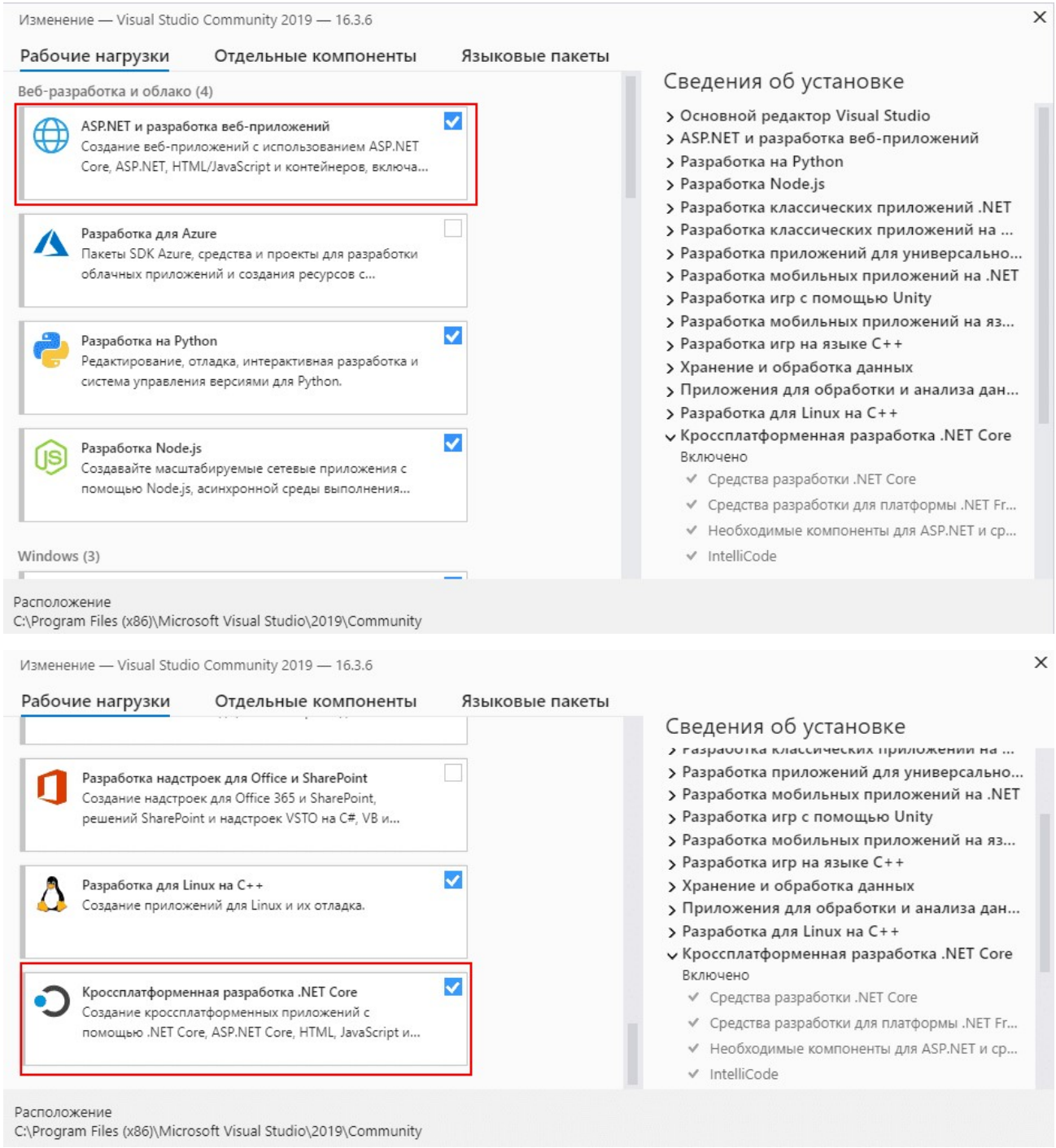


Рисунок 4.2

При створенні проекту виберемо пункт ASP.NET Core Web Application - тип проекту для створення веб-додатки ASP.NET Core (Рисунок4.3).

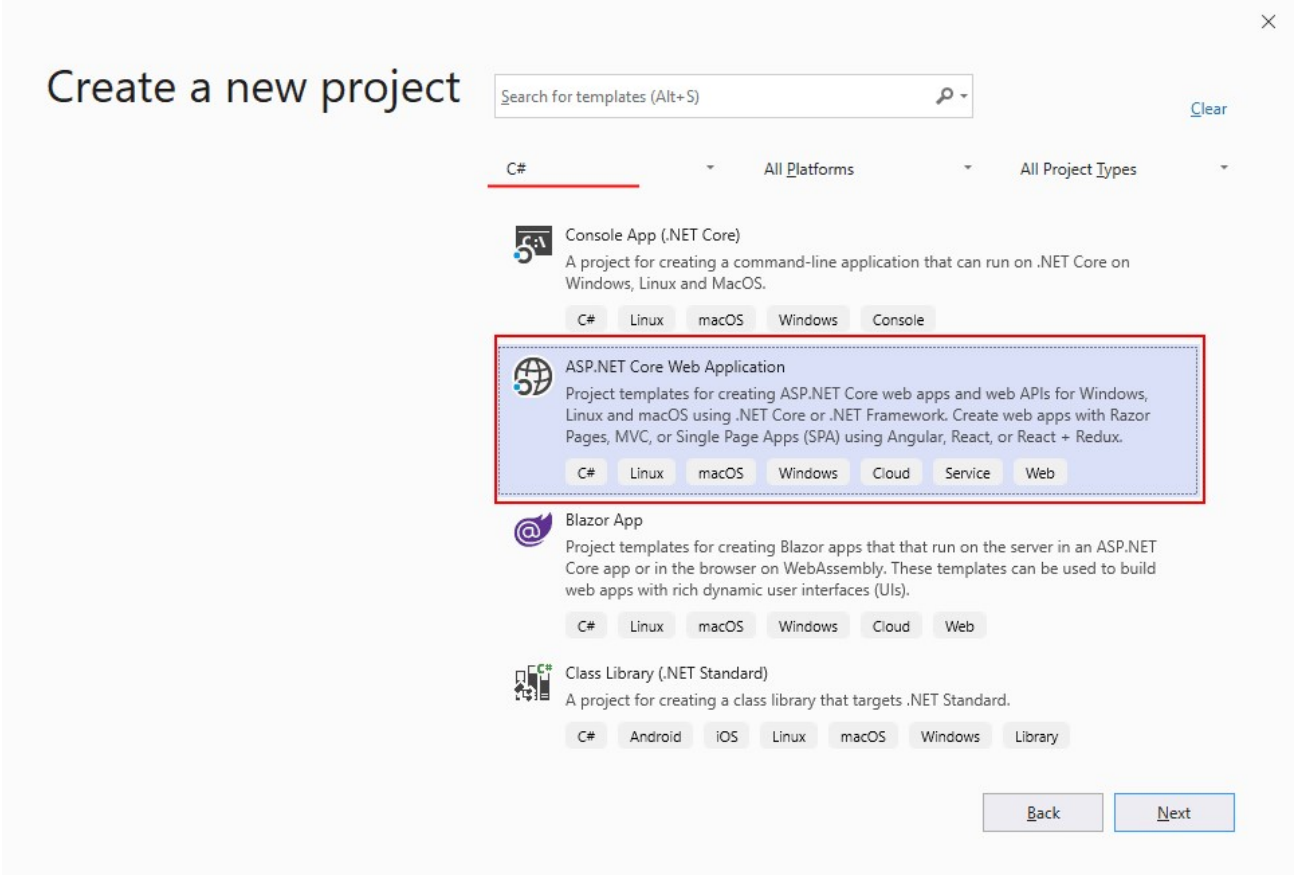


Рисунок 4.3

На наступному кроці вкажемо ім'я проекту і визначимо для нього місце розташування на жорсткому диску (Рисунок4.4).

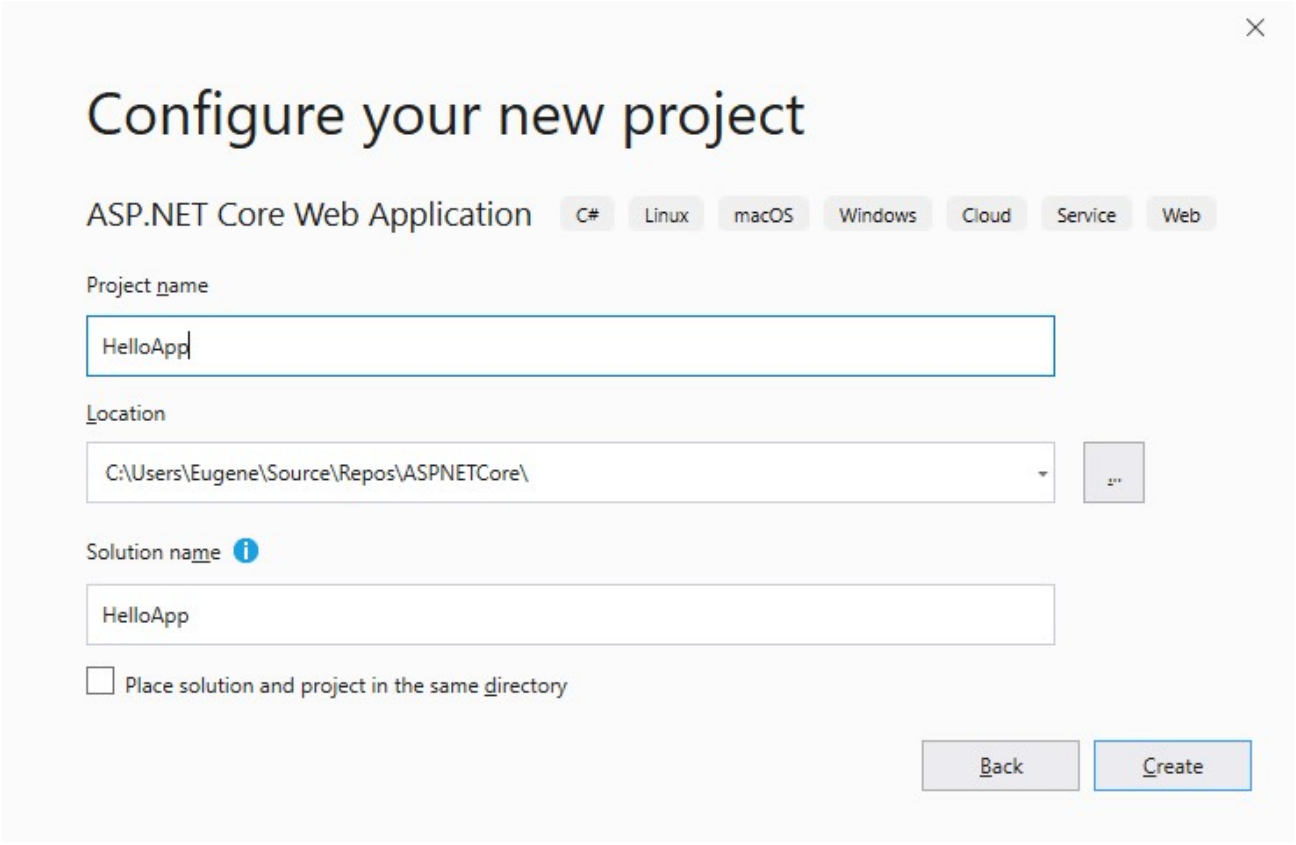


Рисунок 4. 4

Після цього на екрані з'явиться вікно вибору шаблону нової програми (Рисунок4.5):

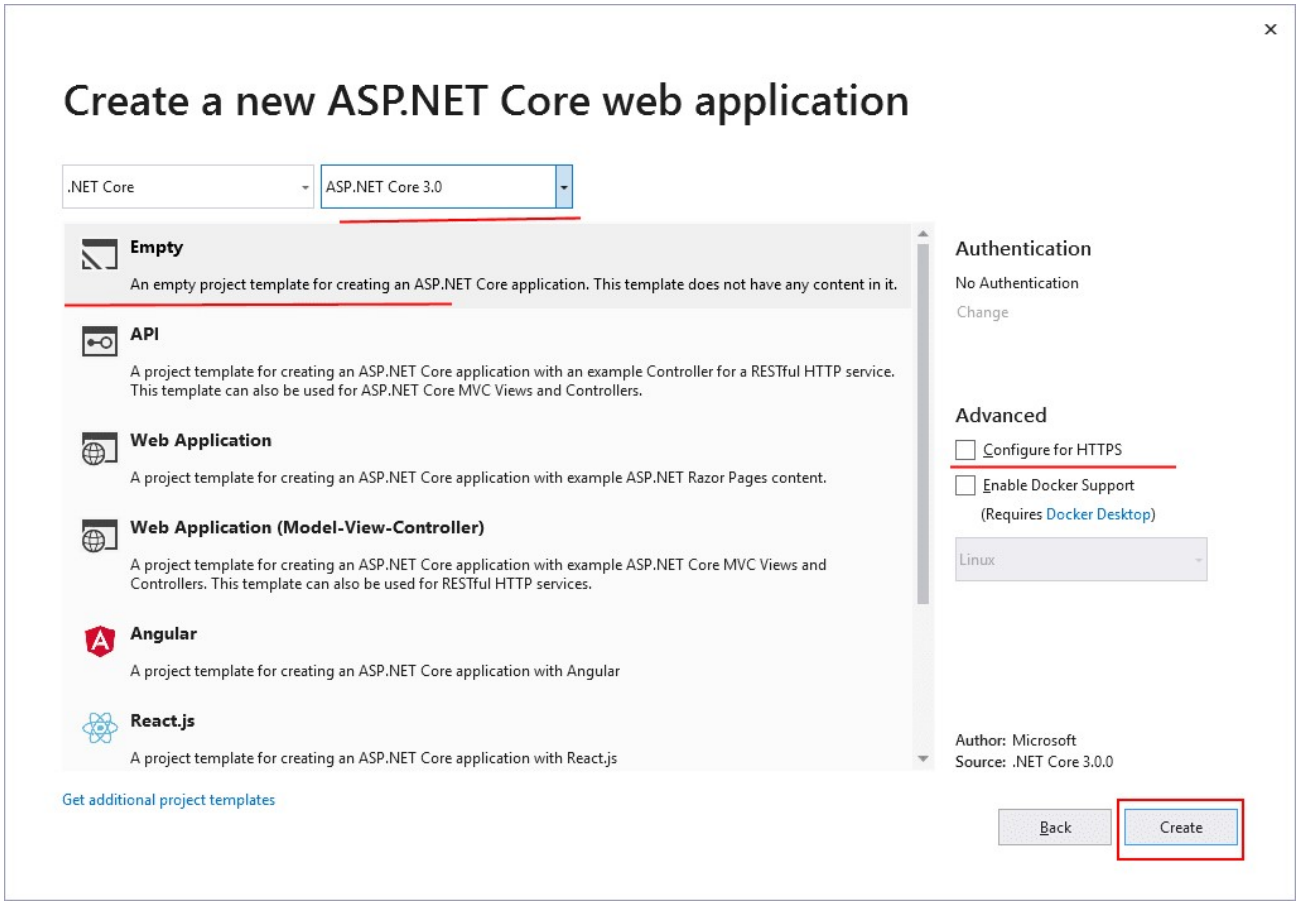


Рисунок 4.5

Вибираємо Empty. Буде створено пустий шаблон з мінімальною функціональністю для створення додатку з нуля [10].

4.2 Налаштування проекту Angular

За замовчуванням новий проект ASP.NET Core не володіє ніякої вбудованої підтримкою роботи з Angular. Один із способів роботи з Angular в рамках проекту ASP.NET Core представляє пакет Microsoft.AspNetCore.SpaServices.Extensions або простіше кажучи SpaServices. Тому спочатку встановимо даний пакет через Nuget (Рисунок4.6)

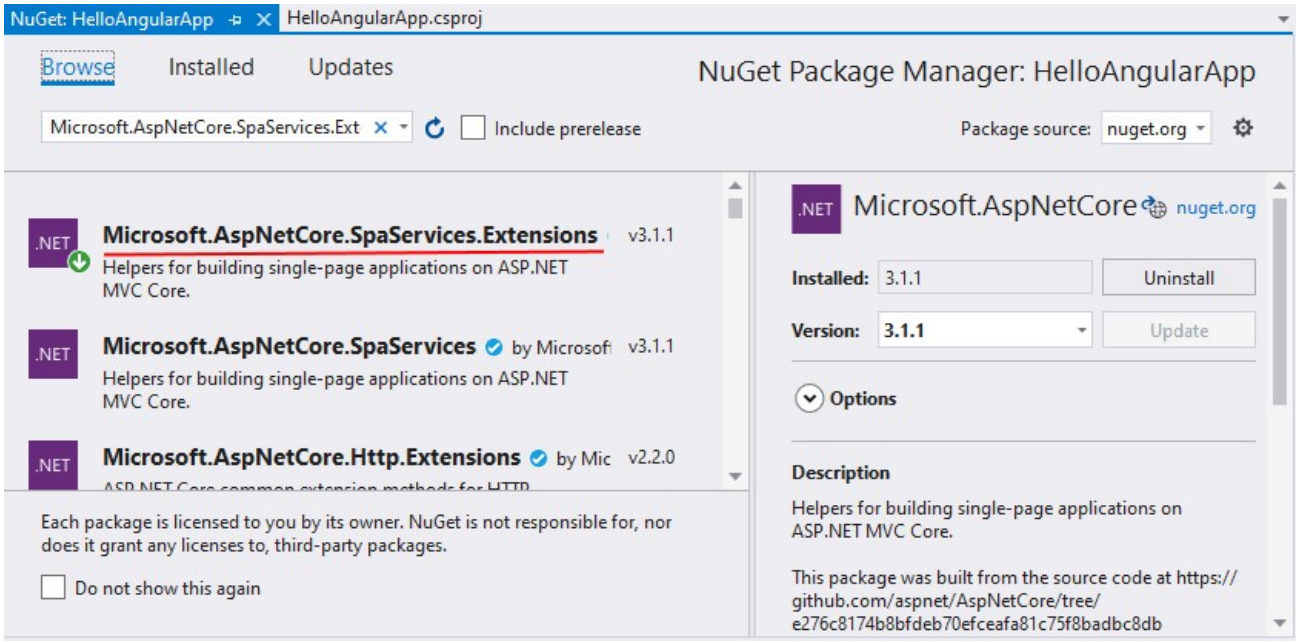


Рисунок 4.6

Після додавання пакета створимо в проекті нову папку, яка буде називатися ClientApp і яка згодом буде зберігати файли і конфігурацію додатка Angular.

Далі додамо в тільки що створену папку ClientApp новий файл package.json, який представляє шаблон npm Configuration file (Рисунок4.7):

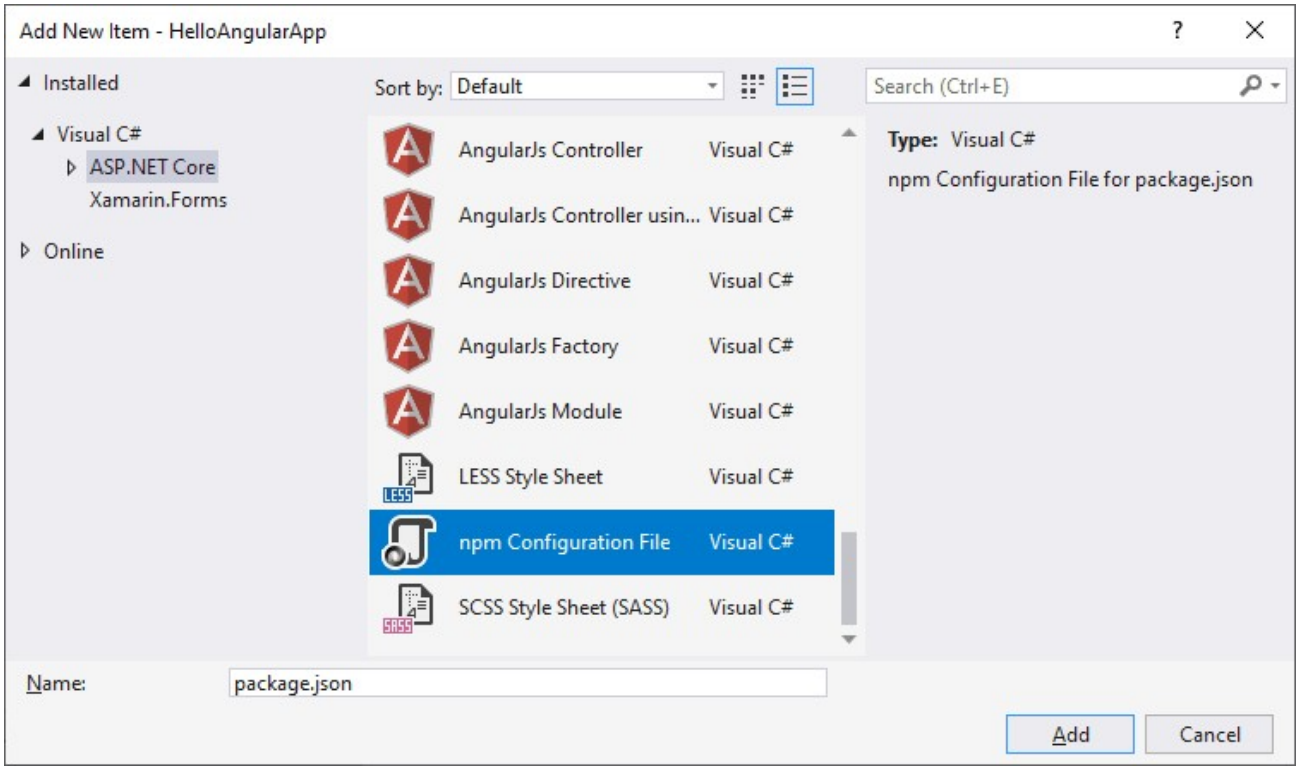


Рисунок 4.7

Змінимо вміст цього файлу наступним чином:

```
{
 "version": "1.1.0",
 "name": "asp.net",
 "scripts": {
  "ng": "ng",
  "start": "echo hello && ng serve",
  "build": "ng build"
 },
 "dependencies": {
  "@angular/common": "~9.0.0",
  "@angular/compiler": "~9.0.0",
  "@angular/core": "~9.0.0",
  "@angular/forms": "~9.0.0",
  "@angular/platform-browser": "~9.0.0",
  "@angular/platform-browser-dynamic": "~9.0.0",
  "@angular/router": "~9.0.0",
  "rxjs": "~6.5.4",
  "zone.js": "~0.10.2"
 },
 "devDependencies": {
  "@angular-devkit/build-angular": "~0.900.1",
  "@angular/cli": "~9.0.1",
  "@angular/compiler-cli": "~9.0.0",
  "@types/node": "^12.11.1",
  "typescript": "~3.7.5"
 }
}
```

Тут визначені всі необхідні пакети фреймворка Angular, а також пакети, які допоможуть при розробці, зокрема, TypeScript і Webpack.

Збережемо файл, і в проект будуть встановлені пакети.

Також додамо в проект файл конфігурації TypeScript tsconfig.json (Рисунок 4.8):

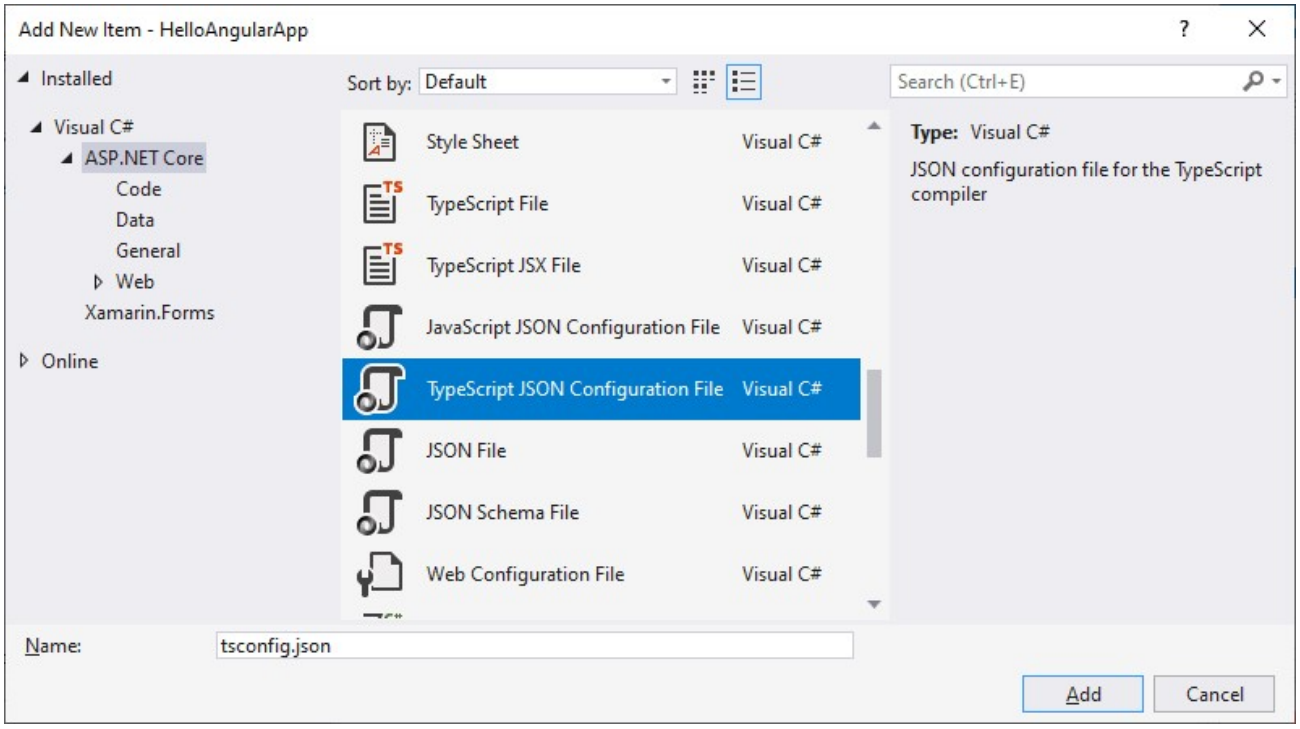


Рисунок 4.8

Після додавання змінимо вміст файлу наступні чином:

```
{
 "compileOnSave": false,
 "compilerOptions": {
  "baseUrl": "./",
  "downlevelIteration": true,
  "experimentalDecorators": true,
  "moduleResolution": "node",
  "target": "es2015",
  "typeRoots": [
   "node_modules/@types"
  ],
  "lib": [
   "dom"
  ]
 },
 "include": [
  "src/**/*.d.ts"
 ]
}
```

Для продовження потрібно в каталозі ClientApp створити нову папку, назвавши її - src, яка буде зберігати файли програми angular. У цій папці створимо

каталог app, в який додамо новий файл app.component.ts і додамо в нього такий вміст:

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

@Component({
   selector: 'app',
   template: `<label>Введи ім’я:</label>
            <input [(ngModel)]="name" placeholder="name">
            <h2>Привіт, {{name}}!</h2>`
})
export class AppComponent {
   name= '';
}
```

Наступним кроком в папці ClientApp/src/app потрібно створити новий файл назвавши його - app.module.ts, який буде представляти головний модуль програми:

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule }   from '@angular/forms';
import { AppComponent }   from './app.component';
@NgModule({
   imports:      [ BrowserModule, FormsModule ],
   declarations: [ AppComponent ],
   bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Наступним кроком в папці ClientApp/src потрібно додати новий файл назвавшм його - main.ts, з якого починається завантаження програми:

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

enableProdMode();
const p = platformBrowserDynamic();
```

```
p.bootstrapModule(AppModule);
```

Крім власне завантаження програми даний файл визначає код для перезапуску програми за допомогою Hot Module при змінах в файлах з вихідним кодом. І також в локацію ClientApp/src потрібно додати файл котрий буде називатися polyfills.ts і це буде виглядати так:

```
import 'zone.js/dist/zone';
```

І також в локацію ClientApp потрібно додати новий файл - angular.json (Рисунок 4.9):

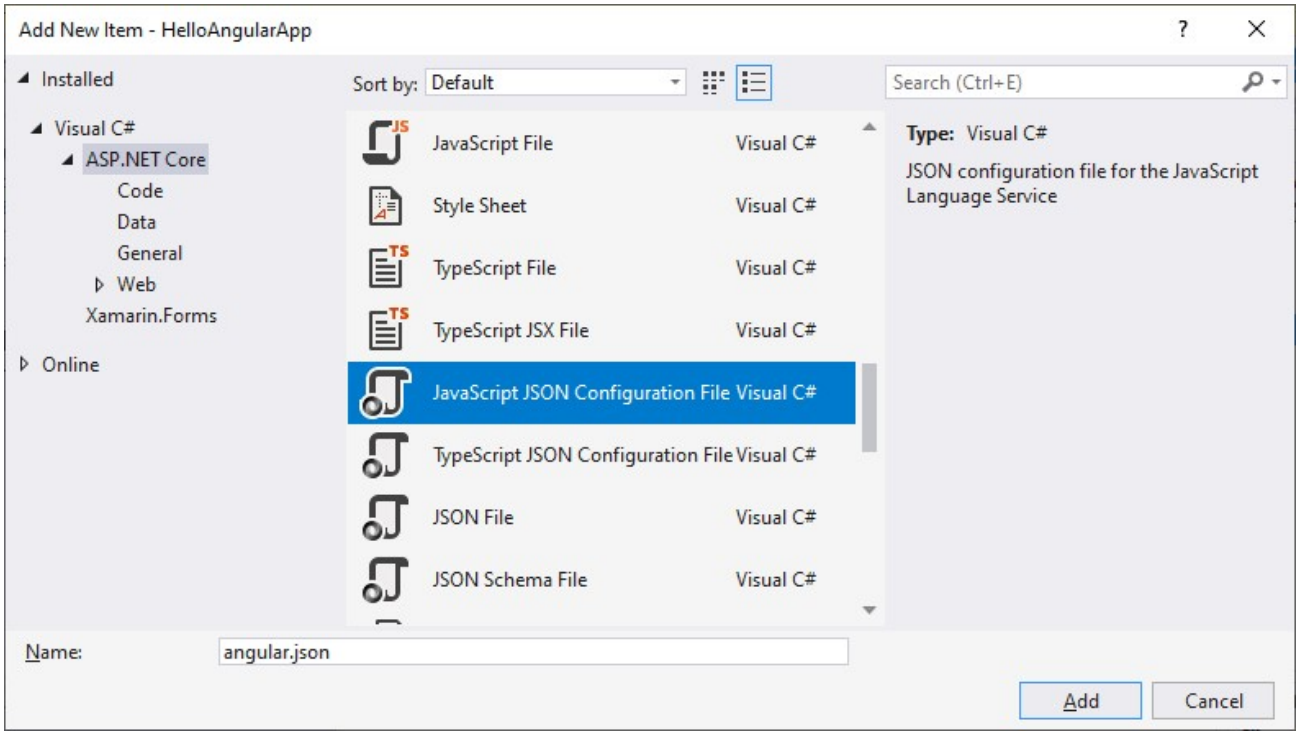


Рисунок 4.9

який буде визначати конфігурацію Angular Cli і виглядає це ось так:

```
{
 "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
 "version": 1,
 "newProjectRoot": "projects",
 "projects": {
  "ToDoList-Angular-JsonSever": {
   "projectType": "application",
   "root": "",
   "sourceRoot": "src",
   "prefix": "app",
```

```
    "architect": {
      "build": {
        "builder": "@angular-devkit/build-angular:browser",
        "options": {
          "outputPath": "dist/ToDoList-Angular-JsonSever",
          "index": "src/index.html",
          "main": "src/main.ts",
          "aot": true,
          "assets": [
            "src/favicon.ico",
            "src/assets"
          ],
          "styles": [
            "src/styles.css"
          ],
          "scripts": []
        },
      },
      "test": {
        "builder": "@angular-devkit/build-angular:karma",
        "options": {
          "main": "src/test.ts",
          "polyfills": "src/polyfills.ts",
          "tsConfig": "tsconfig.spec.json",
          "assets": [
            "src/favicon.ico",
            "src/assets"
          ],
          "styles": [
            "src/styles.css"
          ],
          "scripts": []
        }
      },
    }
  }},
  "defaultProject": "ToDoList-Angular-JsonSever"
}
```

Таким чином, всі файли будуть компілюватися в дві збірки, а саме polyfills.js і app.js, які будуть в папці wwwroot/dist.

Тепер додамо в проект папку ClientApp / src головну веб-сторінку програми - файл index.html і виглядатиме це ось так:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <base href="/" />
  <meta content="width=device-width, initial-scale=1.0" name="viewport" />
  <title>Diploma App</title>
</head>
<body>
  <app>Завантаження...</app>
</body>
</html>
```

В результаті весь проект буде виглядати наступним чином (Рисунок 4.10):

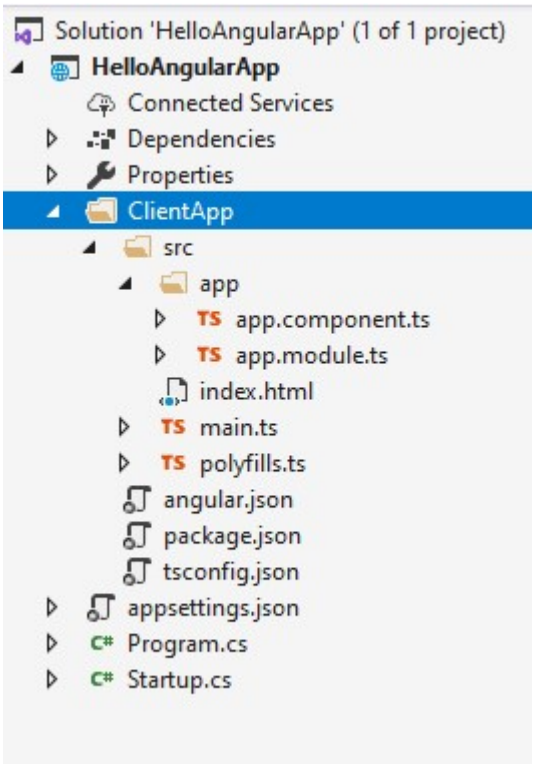


Рисунок 4.10

І наприкінці змінимо Startup.cs файл:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
```

```
using Microsoft.AspNetCore.SpaServices.AngularCli;

namespace DiplomaApp
{
   public class Startup
   {
      public void ConfigureServices(IServiceCollection services)
      {
         services.AddSpaStaticFiles(configuration =>
         {
            configuration.RootPath = "DiplomaApp/dist";
         });
      }

      public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
      {
         if (env.IsDevelopment())
         {
            app.UseDeveloperExceptionPage();
         }
         app.UseStaticFiles();
         if (!env.IsDevelopment())
         {
            app.UseSpaStaticFiles();
         }

         app.UseSpa(spa =>
         {
            spa.Options.SourcePath = "DiplomaApp";

            if (env.IsDevelopment())
            {
               spa.UseAngularCliServer(npmScript: "start");
            }
         });
      }
   }
}
```

В вище вказаному методі - ConfigureServices() ми можемо бачити виклик методу AddSpaStaticFiles(). Він додає в реєстр сервіс ISpaStaticFileProvider. Даний сервіс дозволяє обробляти статичні файли програми Angular з певного місця

розташування. Розташування файлів задається за допомогою параметра configuration.RootPath. В даному випадку це каталог ClientApp / dist, куди, як ми вказали у файлі angular.json, повинні поміщатися скомпільовані файли. Тобто даний сервіс необхідний нам, коли розробка завершена, файли Angular на typescript скомпільовані в файли javascript, а весь додаток ASP.NET Core вже розгорнуто на сервері в режимі production.

Далі в методі Configure () додаються відповідні middleware, які працюють з програмою Angular.

По-перше, якщо розробка закінчена і додаток вже в стані розгортання, викликається метод UseSpaStaticFiles (), який дозволяє направляти запити до нашого додатку Angular.

Далі викликається метод app.UseSpa (), який дозволяє у відповідь на запити відправляти веб-сторінку за замовчуванням (index.html). Цей middleware повинен поміщатися в кінці конвеєра.

У методі app.UseSpa(), якщо додаток знаходиться в режимі розробки, в конвеєр обробки запиту додається spa.UseAngularCliServer компонент. Цей middleware дає можливість налаштувати проект так, щоб ASP.NET Core проект автоматично виконує запуск сервера Angular CLI. А також, UseAngularCliServer додає проксі між сервером додатка Angular і сервером додатка ASP.NET Core. Параметром цей метод приймає команду з файлу package.json. У нашому випадку це команда "start, котра запускає веб-сервер Angular CLI.

Тобто коли ми запустимо ASP.NET Core проект на виконання, то фактично будуть запущені два сервера з двома додатками, які через проксі обмінюватимуться даними. Якщо ми змінимо код в файлах Angular, то моментально станеться перекомпіляція файлів на typescript, і ми побачимо результати наших змін в браузері. Однак якщо ми змінимо код програми ASP.NET Core (тобто код C #), тоді буде потрібно перекомпіляція і перезапуск всієї програми ASP.NET Core, що також перезапустить сервер з додатком Angular.

Що, в кінцевому підсумку, призведе до збільшення часу запуску та деякого уповільнення процесу розробки. Власне це основний мінус UseAngularCliServer. Хоча можна запускати окремо обидва сервера вручну.

4.3 Форматування даних

В цьому розділі описано форматування даних до формату сумісного з нашою системою.

В даному випадку дані приходять у форматі json і повинні бути конвертовані, в об’єкти котрі містяться в нашій системі.

Для збереження даних створена модель InvestigationModel

```
0 references
public class InvestigationModel
{
    0 references
    public int yearOnReceivingData { get; set; }
    0 references
    public List<InvstigationResultItem> ResultItems { get; set; }
}
```

Вона описує дані, котрі ми маємо про певне дослідження. А саме рік дослідження та список результатів. Результати в свою чергу описані власною моделлю InvestigationResultI

```
1 reference
public class InvstigationResultItem
{
    0 references
    public int ResultId { get; set; }
    0 references
    public string Value { get; set; }
    0 references
    public float CountOfVotes { get; set; }
}
```

Вона описує дані одного з результатів дослідження. Тобто дослідження має кілька варіантів відповідей і дана модель описує дані про те, який це саме варіант та скільки голосів було віддано саме за цей варіант.

Для конвертації даних було створено окремий сервіс ParseDataService

```
0 references
public class ParseDataService
{
    0 references
    public static InvestigationModel ParseTargetFile(string path)
    {
        // deserialize JSON directly from a file
        using (StreamReader file = File.OpenText(path))
        {
            JsonSerializer serializer = new JsonSerializer();
            var investigation2 = (InvestigationModel)serializer.Deserialize(file, typeof(InvestigationModel));
            return investigation2;
        }
    }
}
```

Основним завданням цього сервісу є зчитування файлу та десеріалізація цього файлу в потрібні нам об’єкти.

Для безпосереднього аналізу вхідних даних буде використовуватися спеціальний сервіс AnalyzationService. За його допомогою будуть оброблятися вхідні дані і на основі цих даних формуватися звіти для аналізу соціологічних досліджень.

4.4 Реалізація методу кластеризації

Бібліотеки, котрі використовуються в даному проекті є свого роду обгорткою навколо різних модулів. Деякі з них написані на інших мовах програмування. Наприклад метод кластеризації є реалізованим мовою програмування Python. В контексті цієї дипломної роботи, я не вважаю доцільним вказувати цю мову програмування як один з інструментів розробки, оскільки нею було написано окремий модуль, котрий був розроблений поза межами даної дипломної роботи і використовується він через обгортку, котра є сумісною з технологією .NET[16].

Приклад запуску алгоритму не стандартизованих даних:

```
X_std = StandardScaler().fit_transform(df)


km = Kmeans(n_clusters=2, max_iter=100)
km.fit(X_std)
centroids = km.centroids


fig, ax = plt.subplots(figsize=(6, 6))
plt.scatter(X_std[km.labels == 0, 0], X_std[km.labels == 0, 1],
            c='green', label='cluster 1')
plt.scatter(X_std[km.labels == 1, 0], X_std[km.labels == 1, 1],
            c='blue', label='cluster 2')
plt.scatter(centroids[:, 0], centroids[:, 1], marker='*', s=300,
            c='r', label='centroid')
plt.legend()
plt.xlim([-2, 2])
plt.ylim([-2, 2])
plt.xlabel('Eruption time in mins')
plt.ylabel('Waiting time to next eruption')
plt.title('Visualization of clustered data', fontweight='bold')
ax.set_aspect('equal');
```

Приклад ініціалізації центроїдів:

```
n_iter = 9
fig, ax = plt.subplots(3, 3, figsize=(16, 16))
ax = np.ravel(ax)
centers = []
for i in range(n_iter):
    # Run local implementation of kmeans
    km = Kmeans(n_clusters=2,
                max_iter=3,
                random_state=np.random.randint(0, 1000, size=1))
    km.fit(X_std)
    centroids = km.centroids
    centers.append(centroids)
    ax[i].scatter(X_std[km.labels == 0, 0], X_std[km.labels == 0, 1],
                  c='green', label='cluster 1')
    ax[i].scatter(X_std[km.labels == 1, 0], X_std[km.labels == 1, 1],
                  c='blue', label='cluster 2')
    ax[i].scatter(centroids[:, 0], centroids[:, 1],
                  c='r', marker='*', s=300, label='centroid')
    ax[i].set_xlim([-2, 2])
    ax[i].set_ylim([-2, 2])
    ax[i].legend(loc='lower right')
    ax[i].set_title(f'{km.error:.4f}')
    ax[i].set_aspect('equal')
plt.tight_layout();
```

4.5 Реалізація методу регресії

Ще одним методом котрий використовується в даному програмному продукті є метод регресії. На даному зображенні відображена ще одна реалізація мовою Python в одному з модулів, котрі були підключені до нашого проекту[17].

```
import numpy as np

def gradientd():
    param[0][0]=float(param[0][0])-((lrate)*(1/m)*(diff.sum()))
    column=np.zeros(10).reshape(10,1)
    arr = x.tolist()
    for i in range(m):
        i=i+1
        for n in range(m):
            column[n][0]=float(arr[i-1][n])
        param[i][0]=float(param[i][0])-((lrate)*(1/m)*(diff.getT()*column))

def formatprint():
    stuff = ""
    arr = param.tolist()
    for i in range(m+1):
        print('Theta'+str(i)+': '+ str(arr[i][0]))
    print('Cost:', cost)


m=3

x = np.matrix('-25.836 -48.53978 11 1; -25.835727       -48.539798 9 1; -15.819654 -47.946023   1098 1; -15.81876 -47.9
y=np.matrix('0;0;0;0;0;0;1;1;1;1')
param=np.zeros(4).reshape(4,1)
lrate = .000005
for i in range(10000):
    calcy = x*param
    diff = calcy-y
    cost = (1/(2*m))*(float(diff.getT()*diff))
    gradientd()
    formatprint()
    print()
```

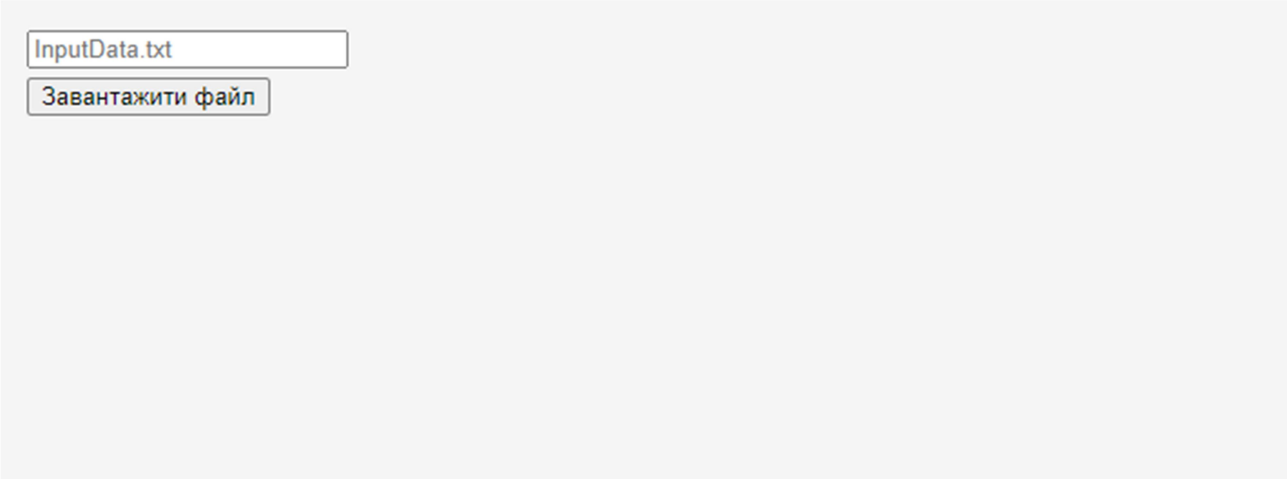
4.6 Приклад роботи

Для роботи з даними нашій системі потрібно, щоб ці дані було попередньо відформатовані. При потребі можна під’єднатися до бази соціологічного центру і написати сервіс, котрий буде парсити дані соціологічного центру і конвертувати їх у відповідний і підходящий формат. Ось приклад даних для нашої програми:

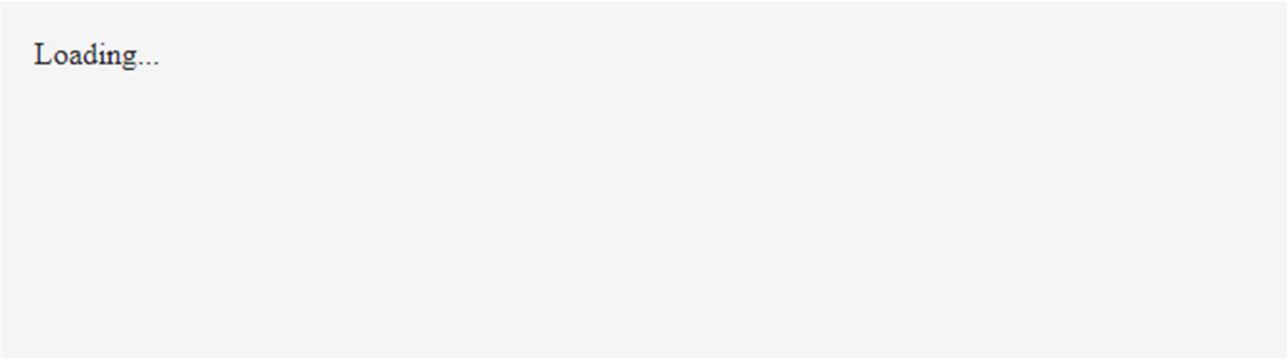
```
C5,13.7659636 4.7984,14 4.55,14 L2.45,14 C2.2016,14 2,13.7659636
2,13.4775273 L2,7.52247273 C2,7.23403636 2.2016,7 2.45,7 L4.55,7 Z
M6.54470232,13.2 C6.24016877,13.1641086 6.01734614,12.8982791
6,12.5737979 C6.01734614,12.5737979 6.01344187,9.66805666 6,8.143986
C6.01344187,7.61903931 6.10849456,6.68623352 6.39801308,6.27384278
C7.10556287,5.26600749 7.60281698,4.6079584 7.89206808,4.22570082
C8.18126341,3.8435016 8.52813047,3.4708734 8.53777961,3.18572676
C8.55077527,2.80206854 8.53655255,2.79471518 8.53777961,2.35555666
C8.53900667,1.91639814 8.74565444,1.5 9.27139313,1.5 C9.52544997,1.5
9.7301456,1.55690094 9.91922413,1.80084547 C10.2223633,2.15596568
10.4343097,2.71884727 10.4343097,3.60971169 C10.4343097,4.50057612
9.50989975,6.1729303 9.50815961,6.18 C9.50815961,6.18
13.5457098,6.17908951 13.5464084,6.18 C14.1635544,6.17587601
14.5,6.72543196 14.5,7.29718426 C14.5,7.83263667 14.1341135,8.278973
13.6539433,8.3540827 C13.9452023,8.49286263 14.1544715,8.82364675
14.1544715,9.20555417 C14.1544715,9.68159617 13.8293011,10.0782687
13.3983805,10.1458495 C13.6304619,10.2907572 13.7736931,10.5516845
13.7736931,10.847511 C13.7736931,11.2459343 13.5138356,11.5808619
13.1594388,11.6612236 C13.3701582,11.7991865 13.5063617,12.0543945
```

Цей набір даних парситься нашою програмою і на основі цих даних генерується прогноз на наступний період часу. Після відпрацювання усієї логіки по аналізу і прогнозуванню, починається робота над побудовою графіка, тобто візуального представлення згенерованого прогнозу. Таким чином переходимо до графічного інтерфейсу.

Першим що бачить користувач є ось такий стартовий екран:



Тут користувач натискає на кнопку і завантажує файл з відформатованими даними. Поки триває обробка інформації, користувач бачить таку сторінку:

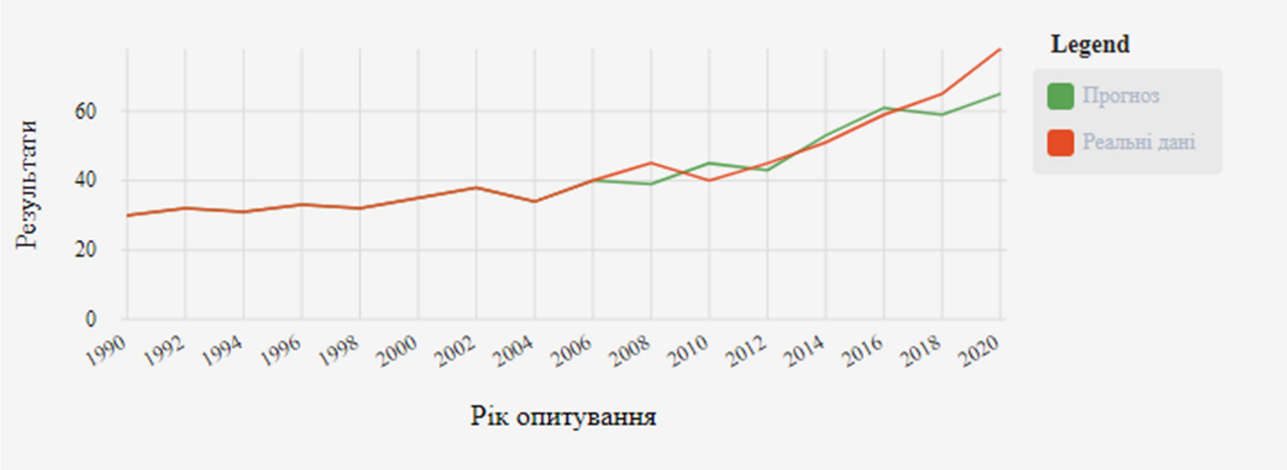


Користувацький інтерфейс зроблений за допомогою фреймворку Angular. Це дуже потужний і популярний фреймворк, котрий дозволяє створювати односторінкові веб-сайти (Single Page Applications). В нашому випадку його більше, ніж достатньо. Для відображення графіків в цьому проекті використовується npm пакет - @swimlane/ngx-charts. Це унікальний пакет, тому що він не просто обертаємо d3 або будь-який інший механізм діаграм. Він використовує Angular для візуалізації та анімації елементів SVG з усіма перевагами зв'язування та швидкості, а також використовує d3 для чудових математичних функцій, масштабів, генераторів осей і форм. Завдяки тому, що Angular виконує весь рендеринг, цей пакет відкриває нам безмежні можливості, які надає платформа Angular. Ось приклад того як виглядає код побудови графіка:

```
  showLabels: boolean = true;
  animations: boolean = true;
  xAxis: boolean = true;
  yAxis: boolean = true;
  showYAxisLabel: boolean = true;
  showXAxisLabel: boolean = true;
  xAxisLabel: string = 'Year';
  yAxisLabel: string = 'Answers';
  timeline: boolean = true;

  colorScheme = {
    domain: ['#5AA454', '#E44D25', '#CFC0BB', '#7aa3e5', '#a8385d', '#aae3f5'],
  };

  constructor() {
    Object.assign(this, { multi });
  }

  onSelect(data): void {
    console.log('Item clicked', JSON.parse(JSON.stringify(data)));
  }

  onActivate(data): void {
    console.log('Activate', JSON.parse(JSON.stringify(data)));
  }

  onDeactivate(data): void {
    console.log('Deactivate', JSON.parse(JSON.stringify(data)));
  }
}
```

```
<ngx-charts-line-chart
  [view]="view"
  [scheme]="colorScheme"
  [legend]="legend"
  [showXAxisLabel]="showXAxisLabel"
  [showYAxisLabel]="showYAxisLabel"
  [xAxis]="xAxis"
  [yAxis]="yAxis"
  [xAxisLabel]="xAxisLabel"
  [yAxisLabel]="yAxisLabel"
  [timeline]="timeline"
  [results]="multi"
  (select)="onSelect($event)"
  (activate)="onActivate($event)"
  (deactivate)="onDeactivate($event)"
  >
</ngx-charts-line-chart>
```

Після аналізу вхідних даних, дана програма створює прогноз і надсилає дані користувацькому інтерфейсу. Після цього відбувається побудова графіка і користувач бачить на своєму екрані ось такий графік:



Коротко про те що тут показано. На даному графіку показано дві лінії: прогноз і реальні дані. Це дані по опитуванню стосовно глобального потепління. Результатом цього опитування є відсоток людей, котрі вважають питання глобального потепління вкрай важливим. На даний момент є тут присутні дані з 1990 року по 2020 рік. З метою перевірки результатів роботи даної системи для аналізу було передано дані до 2006 року. Таким чином ми можемо отримати результат аналізу і прогнозу від системи і порівняти його з реальними результатами опитування. Як бачимо прогноз не зовсім співпадає з реальними даними. Для людей питання глобального потепління виявилося більш критичним, ніж того очікувала наша система. Але варто зауважити, що

було прогнозовано ріст актуальності цього питання і цей ріст все таки відбувся, але в трохи більшій мірі, ніж прогнозовано.

Висновки до розділу

Новостворене програмне забезпечення має доволі потужну основу, котра будована на останніх технологіях і по великій мірі не містить застарілого коду чи бібліотек. Тут використовується програмне забезпечення та методи розробки з різних сфер інформаційних технологій. Наприклад, Angular є веб-фреймворком і призначений тільки для веб-розробки. .NET є широко застосовуваною технологією і буде важко перерахувати сфери в яких він використовується та типи програмного забезпечення, котре може бути створене з його допомогою, оскільки він є кросплатформенним і доволі продуктивним. Алгоритми та методи, котрі використовуються для аналізу даних є доволі поширеними і продуктивними в обробці даних. Беручи до уваги перераховані вище технології, можна зробити висновок, що проблем в цього проекту зі сторони програмного забезпечення, бути не повинно і при правильному маркетингу та менеджменту у цього програмного забезпечення є всі шанси стати успішним та популярним продуктом на ринку.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Стартап - це компанія, яка знаходиться на початкових етапах бізнесу. Засновники зазвичай фінансують свої стартапи і можуть спробувати залучити зовнішні інвестиції, перш ніж вони запустять. Джерела фінансування включають сім'ю та друзів, венчурних капіталістів, краудфандинг та позики.

Як випливає з цього терміну, «Startup» не є постійною фазою для будь-якого бізнесу, і він не стосується лише компаній у сфері технологій. Це життєво важливий, ранній етап життєвого циклу бізнесу, і він може стосуватися практично будь-якої галузі.

Загалом, у стартапів, як правило, мало працівників і потенціал швидкого зростання. Вони забезпечують широку привабливість продуктів, які або ще не існують, або вирішують проблему краще, ніж доступні на даний момент варіанти.

«Що таке стартап?» це не таке просте питання, як може здатися. Визначення стартапу змінюється залежно від того, кого ви запитаєте, хоча є деякі загальні характеристики, які застосовуються в усьому світі.

Взагалі кажучи, якщо компанія існує більше кількох років, має більше кількох співробітників або отримує багатомільйонний дохід, вона, ймовірно, виросла після фази запуску.

Даний програмний продукт може бути реалізований у вигляді стартапу. На даний момент у нього є потенціал, бо є чимало організацій котрі займають аналітикою і проведенням опитувань, але не прогнозуванням майбутніх результатів. Основна ідея даного проекту полягає в тому щоб прогнозувати важливі соціальні опитування. При умові вдосконалення машинного навчання та збільшення обсягу даних котрі зможе обробляти дана програма, у цього проекту є хороший потенціал.

Можливі напрямки застосування:

- Прогнозування результатів соціологічних досліджень
- Аналіз уже існуючих досліджень
- Виявлення недостовірних опитувань на основі статистичних даних

Основною вигодою від використання цього програмного продукту є те, що можна передбачати настрої серед населення і на основі цього виконувати якісь дії, а також аналізуючи результати вже існуючих досліджень, ми можемо визначати недобросовісних опитувальників і вже на основі цього робити те, що вважаємо за потрібне: просто не брати до уваги результати опитувань цього опитувальника, а бо й взагалі повідомляти громадськість про ситуацію яка склалася і попереджати про те, що варто відноситися з обережність до результатів роботи даної організації. Також можна провести перемовини з державними органами і домовитися про співпрацю, основною ціллю якої буде виявлення недобросовісних опитувальників і застереження громадян. Я вважаю. що такий розвиток подій має сенс. Якщо домовитися про вигідну фінансову складову, то для мене як для власника даного продукту буде поєднуватися приємне з корисним. Тобто покращення фінансового стану з позитивною активністю у громадському секторі (запобігання поширенню недостовірної і недобросовісної інформації).

Даний програмний продукт може стати прибутковим і приносити непоганий дохід, але цільовою аудиторією, тобто тими хто буде платити гроші буде не фізичні особи, а держані органи, громадські організації або й комерційний сектор. Фактично для пересічного громадянина цей програмний продукт не нестиме ніякої цінності. Важливим потенційним клієнтом є комерційний сектор. Як приклад: компанії, котрі зацікавлені в громадській активності та ініціативі. У такому випадку компанія може стати спонсором нашого програмного продукту і в подальшому позиціонувати себе як компанію, котра виступає за чесні і прозорі соціологічні опитування і за донесення достовірної інформації до населення.

5.2 Розроблення ринкової стратегії

Маркетинг – це динамічна сфера, яка постійно змінюється. Найпопулярніші стратегії цифрового маркетингу змінюються разом із споживачами та технологічними тенденціями дня. Ось чому кожній компанії потрібна хороша маркетингова стратегія, яка добре спланована та має чітко визначені віхи та цілі. Якщо ви маєте правильну карту, шанси на досягнення цілей, які ви поставили перед своїм бізнесом, значно вищі.

Це означає, що, хоча більшість із нас прагне розпочати маркетингові зусилля відразу після того, як ми вирішимо розпочати бізнес, нам насправді потрібно інвестувати в планування, щоб не витрачати свій обмежений бюджет та енергію на неправильні речі.

Я вважаю, що маркетингова стратегія цього програмного продукту повинна реалізовуватися за певний період часу до того як програмний продукт стане доступним для придбання або для інвестування. Якщо коротко я маю на увазі, що для реалізації цього доволі специфічного проекту потрібно закладати певний фундамент і створити такі умови, щоб на ринку почало бракувати саме такого програмного продукту, котрий повинен бути випущений нами найближчим часом. Що мається на увазі? Якщо просто і коротко, то треба спочатку створити такі умови на ринку, щоб з'явився попит і одразу після цього стати першим на ринку, хто зробить пропозицію, котра задовільний цей попит. У нашому випадку потрібно запровадити кілька громадських і соціальних ініціатив, котрі фокусуватимуть увагу громадськості на тому, що зараз є багато соціальних досліджень і опитувань з недостовірними результатами. Що ймовірність фальсифікації надзвичайно висока і що на даний момент є багато дивних опитувань, котрі не відповідають законам логіки і аналітики, тобто результати з часом змінилися так як не могли б змінитися навіть теоретично. В таких випадку явно прослідковується слід фальсифікації. І потім наштовхуємо людей на думку про те що така проблема є доволі поширеною і з цим потрібно щось робити, бо викривлення даних дає простір для маніпуляцій і створення хибного враження про відношення суспільства до певних важливих соціальних питань, що були порушені і начебто досліджені опитувальниками.

5.3 Вимоги до технічного та програмного забезпечення

Створене програмне забезпечення повинне відповідати певним вимогам. Загалом даний проект має непоганий потенціал, але дуже багато тут залежить від маркетингу. Звичайно не менш важливою є і технічно-програмна сторона реалізації. Загалом є кілька вимог до програмного продукту, котрі повинні бути

реалізовані, або такі котрі можна віднести в категорію «бажані» і реалізувати в наступних ітераціях розробки.

З точки зору технічного забезпечення, то потрібно певні технічні засоби для розробки і виконання програми. Основна робота нашого проекту це аналіз даних. Для точнішого результату потрібно великий об'єм даних на основі котрих алгоритм буде проводити аналіз майбутніх досліджень. Відповідно для обробки великого масиву даних потрібно буде або дуже багато часу або дуже багато ресурсів. Відношення часу і ресурсів у цьому випадку є оберненопропорційним, тобто чим більше ресурсів ми залучимо тим менше часу потрібно буде для обробки даних і навчання системи. І навпаки: чим менше ресурсів буде доступно, тим більше часу займе навчання системи. Хорошим варіантом в даному випадку є використання хмарних сервісів таких як Azure. Вони надають потрібну кількість ресурсів за помірною ціною. І порівнянні з локальною машиною, хмарні сервіси мають чимало плюсів. І якщо їх проаналізувати, то навряд у вас виникне бажання купляти локальну машину і користуватися обмеженою кількістю ресурсів. Плюсом такого використання ресурсів є те, що ви платите за них тільки тоді коли вони вам потрібні. Тобто ви можете скасувати підписку і не платити тоді коли ресурси вам вже не потрібні. Якщо використовувати описаний варіант технічного забезпечення, то ви у будь-якому випадку зможете задовільнити технічні вимоги даного продукту.

Стосовно вимог до програмного забезпечення, то єдиним пунктом, котрий може викликати хвилювання є сумісність з операційною системою машини на котрій буде запущений проект. Основною рекомендацією в даному випадку є використання технології та інструменту Docker. За допомогою цього ви можете упакувати свій програмний продукт в контейнер і запускати його згодом не хвилюючись за операційну систему і сумісність. В даному випадку ваша програма буде запускатися ізольовано і образ упакованої програми буде містити в собі усі необхідні компоненти для запуску.

Висновки до розділу

Даний програмний продукт має непоганий потенціал і може бути реалізований як повноцінний сервіс. Він має кілька варіантів просування в маси і створення попиту для його використання. Як і в більшості випадків, економічний успіх даного проекту по великій мірі залежить від грамотного маркетингу і підготовки ринку, до випуску такого специфічного продукту. Незважаючи ні на що цей проект має шанси бути економічно успішним. Він містить складову, котра може викликати інтерес як в громадськості так і в бізнесу, тому впевнено можна сказати, що потенціал тут присутній.

ВИСНОВКИ

В процесі створення сервісу був проведений аналіз існуючих соціологічних досліджень і оброблені дані результатів цих досліджень. Була узгоджена робота програмних компонентів, розроблений користувацький інтерфейс та весь необхідний функціонал продукту. Це було зроблено за допомогою мов програмування C# та TypeScript. Клієнтська частина додатку була створена за допомогою фреймворку Angular, а серверна частина за допомогою платформи .Net Core.

Дана система була створена з метою обробки результатів соціологічних досліджень і створення звітів на основі проведеного аналізу. Таким чином забезпечується повніше висвітлення ситуації чи теми стосовно, котрої було проведено дане соціологічне дослідження.

Створена система повинна покращити ефективність проведення соціологічних досліджень та дозволить виявити більше певних тенденцій і напрямків розвитку певних соціальних ситуацій.

Результатом цієї роботи є система, котра оброляє дані соціологічних досліджень, аналізує їх і на основі пророблені роботи видає результати. З нашої точки зору це є цілком безкоштовний програмний продукт і ми можемо виокристовувати його невизначений термін нічого не оплачуючи, оскільки даний програмний продукт створений нами і є цілком нашою власністю. Також використовуючи маркетинг та умови ринку ми можемо не тільки економити на тому, що не купляємо подібного програмного забезпечення, а й отримувати прибуток від того, що продаватимемо новостворений продукт клієнтам, котрі зацікавлені в такого роду проектах.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ПОСИЛАНЬ

1. http://uk.wikipedia.org [Електронний ресурс]. – доступний станом на 20.05.2020.
2. https://docs.microsoft.com/ru-ru/ [Електронний ресурс]. – доступний станом на 20.05.2020.
3. https://metanit.com/ [Електронний ресурс]. – доступний станом на 20.05.2020.
4. https://angular.io/ [Електронний ресурс]. – доступний станом на 20.05.2020.
5. https://www.google.com/ [Електронний ресурс]. – доступний станом на 20.05.2020.
6. https://www.tutorialsteacher.com/ [Електронний ресурс]. – доступний станом на 20.05.2020.
7. https://www.c-sharpcorner.com/ [Електронний ресурс]. – доступний станом на 20.05.2020.
8. https://stackoverflow.com/ [Електронний ресурс]. – доступний станом на 20.05.2020.
9. https://habr.com/en/ [Електронний ресурс]. – доступний станом на 20.05.2020.
10. https://www.reddit.com/ [Електронний ресурс]. – доступний станом на 20.05.2020.
11. Джеймс Скотт, Нік Полсон. «AIQ» — Transworld Digital, 2019, 272ст.
12. Пітер Норвіг, Стюарт Рассел. «Штучний інтелект: сучасний підхід» — Prentice Hall, 2009, 1152ст.
13. Том Мітчел, «Машинне навчання» — McGraw-Hill Education, 1997, 414ст.
14. Ian Goodfellow, Yoshua Bengio, Aaron Courville «Deep Learning» — 2005, 800ст.
15. Гудфеллоу Я, Беджіо І, Курвіль А, «Глибинне навчання» — ДМК-Пресс, 2018, 652ст.
16. Франсуа Шолле, «Глибинне навчання з Пайтон» — Hanning, 2018, 400ст.
17. Орельєн Жерон, «Прикладне машинне навчання за допомогою Scikit-Learn і TensorFlow», — Діалектика, 2020, 1040ст.
18. Річард Саттон, Ендрю Барто, «Навчання з підкріпленням» — ДМК Пресс, 2020, 552ст.
19. Ендрю Траск, «Грокаємо машинне навчання» — PaperBack, 2020, 392ст.
20. Джуда Перл, «Книга Чому» — Penguin, 2020, 432ст.
21. Христоф Молнар, «Інтерпретоване машинне навчання» — lulu.com, 2020, 318ст.

ДОДАТКИ

Лістинг програмного коду

ParseDataService.cs

```
public class ParseDataService
{
    public static InvestigationModel ParseTargetFile(string path)
    {
        // deserialize JSON directly from a file
        using (StreamReader file = File.OpenText(path))
        {
            JsonSerializer serializer = new JsonSerializer();
            var investigation2 = (InvestigationModel)serializer.Deserialize(file, typeof(InvestigationModel));
            return investigation2;
        }
    }
}
```

AnalyzationService.cs

```
public class AnalyzationService
{
    public int FindIndexOfSpecificInvestigation(InvestigationModel[] sortedData, InvestigationModel item)
    {
        var leftIndex = 0;
        var rightIndex = sortedData.Length - 1;

        while (leftIndex <= rightIndex)
        {
            var middleIndex = leftIndex + (rightIndex - leftIndex) / 2;

            if (item.CompareTo(sortedData[middleIndex]) > 0)
            {
                leftIndex = middleIndex + 1;
                continue;
            }

            if (item.CompareTo(sortedData[middleIndex]) < 0)
            {
                rightIndex = middleIndex - 1;
                continue;
            }

            return middleIndex;
        }

        return -1;
    }

    private int SelectTheMostEffectiveItem(IList<InvestigationModel> collection, InvestigationModel item, int leftIndex, int rightIndex)
    {
        if (leftIndex > rightIndex)
        {
            return -1;
        }

        var middleIndex = leftIndex + (rightIndex - leftIndex) / 2;
        var result = item.CompareTo(collection[middleIndex]);

        return result switch
        {
            var r when r == 0 => middleIndex,
            var r when r > 0 => SelectTheMostEffectiveItem(collection, item, middleIndex + 1, rightIndex),
            var r when r < 0 => SelectTheMostEffectiveItem(collection, item, leftIndex, middleIndex - 1),
```

```
                _ => -1
            };
        }

        private static double[,] TabulateResultOfAnalization(InvestigationModel[] items, Func<InvestigationModel, int> size, Func<InvestigationModel, double> valueSelector, int maxCapacity)
        {
            var n = items.Length;
            var results = new double[n + 1, maxCapacity + 1];
            for (var i = 0; i <= n; i++)
            {
                for (var w = 0; w <= maxCapacity; w++)
                {
                    if (i == 0 || w == 0)
                    {
                        results[i, w] = 0;
                    }
                    else if (size(items[i - 1]) <= w)
                    {
                        var iut = items[i - 1];
                        var vut = valueSelector(iut);
                        var wut = size(iut);
                        var valueIfTaken = vut + results[i - 1, w - wut];
                        var valueIfNotTaken = results[i - 1, w];
                        results[i, w] = Math.Max(valueIfTaken, valueIfNotTaken);
                    }
                    else
                    {
                        results[i, w] = results[i - 1, w];
                    }
                }
            }

            return results;
        }

        private static InvestigationModel[] GetOptimalItems(InvestigationModel[] items, Func<InvestigationModel, int> size, double[,] cache, int capacity)
        {
            var currentCapacity = capacity;

            var result = new List<InvestigationModel>();
            for (var i = items.Length - 1; i >= 0; i--)
            {
                if (cache[i + 1, currentCapacity] > cache[i, currentCapacity])
                {
                    var item = items[i];
                    result.Add(item);
                    currentCapacity -= size(item);
                }
            }
            result.Reverse();
            return result.ToArray();
        }
    }
```

InvestigationModel.cs

```
    public class InvestigationModel : IComparable
    {
        public int yearOnReceivingData { get; set; }
        public List<InvstigationResultItem> ResultItems { get; set; }

        public int CompareTo(object obj)
        {
```

```
            throw new NotImplementedException();
        }
    }
```

InvstigationResultItem.cs

```
    public class InvstigationResultItem
    {
        public int ResultId { get; set; }
        public string Value { get; set; }
        public float CountOfVotes { get; set; }
    }
```

ParseDataService.cs

```
        public static InvestigationModel ParseTargetFile(string path)
        {
            // deserialize JSON directly from a file
            using (StreamReader file = File.OpenText(path))
            {
                JsonSerializer serializer = new JsonSerializer();
                var investigation2 = (InvestigationModel)serializer.Deserialize(file, typeof(InvestigationModel));
                return investigation2;
            }
        }
```

Startup.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.SpaServices.AngularCli;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews();
            services.AddHttpContextAccessor();
            services.AddCors(o => o.AddPolicy("MyPolicy", builder => {
                builder.AllowAnyOrigin()
                    .AllowAnyMethod()
                    .AllowAnyHeader();

            }));
            services.AddControllers().AddNewtonsoftJson(options =>
            options.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore
            );

            services.AddSpaStaticFiles(configuration =>
            {
                configuration.RootPath = "ClientApp/dist";
            });
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
```

```
            }
            else
            {
                app.UseExceptionHandler("/Error");
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();
            if (!env.IsDevelopment())
            {
                app.UseSpaStaticFiles();
            }

            app.UseRouting();
            app.UseCors("MyPolicy");
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller}/{action=Index}/{id?}");
            });

        }
    }
```

Angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "InternetStore": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "prefix": "app",
      "schematics": {},
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "progress": false,
            "extractCss": true,
            "outputPath": "dist",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "src/tsconfig.app.json",
            "assets": [
              "src/assets"
            ],
            "styles": [
              "src/custom-theme.scss",
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "src/styles.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
              "fileReplacements": [
                {
                  "replace": "src/environments/environment.ts",
                  "with": "src/environments/environment.prod.ts"
```

```
        }
      ],
      "optimization": true,
      "outputHashing": "all",
      "sourceMap": false,
      "extractCss": true,
      "namedChunks": false,
      "aot": true,
      "extractLicenses": true,
      "vendorChunk": false,
      "buildOptimizer": true
    }
   }
},
"serve": {
  "builder": "@angular-devkit/build-angular:dev-server",
  "options": {
    "browserTarget": " AnalyticalSystem:build"
  },
  "configurations": {
    "production": {
      "browserTarget": " AnalyticalSystem:build:production"
    }
  }
},
"extract-i18n": {
  "builder": "@angular-devkit/build-angular:extract-i18n",
  "options": {
    "browserTarget": " AnalyticalSystem:build"
  }
},
"test": {
  "builder": "@angular-devkit/build-angular:karma",
  "options": {
    "main": "src/test.ts",
    "polyfills": "src/polyfills.ts",
    "tsConfig": "src/tsconfig.spec.json",
    "karmaConfig": "src/karma.conf.js",
    "styles": [
      "./node_modules/@angular/material/prebuilt-themes/purple-green.css",
      "src/styles.css"
    ],
    "scripts": [],
    "assets": [
      "src/assets"
    ]
  }
},
"lint": {
  "builder": "@angular-devkit/build-angular:tslint",
  "options": {
    "tsConfig": [
      "src/tsconfig.app.json",
      "src/tsconfig.spec.json"
    ],
    "exclude": [
      "**/node_modules/**"
    ]
  }
},
"server": {
  "builder": "@angular-devkit/build-angular:server",
  "options": {
    "outputPath": "dist-server",
    "main": "src/main.ts",
```

```
          "tsConfig": "src/tsconfig.server.json"
        },
        "configurations": {
          "dev": {
            "optimization": true,
            "outputHashing": "all",
            "sourceMap": false,
            "namedChunks": false,
            "extractLicenses": true,
            "vendorChunk": true
          },
          "production": {
            "optimization": true,
            "outputHashing": "all",
            "sourceMap": false,
            "namedChunks": false,
            "extractLicenses": true,
            "vendorChunk": false
          }
        }
      }
    },
    "InternetStore-e2e": {
      "root": "e2e/",
      "projectType": "application",
      "architect": {
        "e2e": {
          "builder": "@angular-devkit/build-angular:protractor",
          "options": {
            "protractorConfig": "e2e/protractor.conf.js",
            "devServerTarget": "InternetStore:serve"
          }
        },
        "lint": {
          "builder": "@angular-devkit/build-angular:tslint",
          "options": {
            "tsConfig": "e2e/tsconfig.e2e.json",
            "exclude": [
              "**/node_modules/**"
            ]
          }
        }
      }
    }
  },
  "defaultProject": " AnalyticalSystem"
}
```