ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра системного аналізу

**Пояснювальна записка**

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: **«Інформаційна система прийняття рішень при розгляданні заявок на відновлення власності громадян України»**

Виконала: студент 4 курсу,
групи САД–41 спеціальності
124 Системного аналізу_____
(шифр і назва спеціальності)

Литвиненко О.В._____
(прізвище та ініціали)

Керівник Козаченко С.Я._____
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Київ – 2023

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**Навчально-науковий інститут телекомунікацій**

Кафедра Системного аналізу _____
Ступінь вищої освіти - «Бакалавр»_____
Спеціальність - 124 Системний аналіз_____

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Системного аналізу
_____Гордієнко Т.Б

“____” _____2023 року

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ**

_____Литвиненка Олексія Вікторовича_____
(прізвище, ім‘я, по батькові)

1.Тема роботи:
Інформаційна система прийняття рішень при розгляданні заявок на **відновлення** власності громадян України
Керівник роботи____Козаченко С.Я._____,
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “ “_____2023 року №__.
2. Строк подання студентом роботи_____
3. Вхідні дані до роботи:
Література з питань роботи СППР(DSS);
Інструменти розробки, мова програмування Python, база даних PostgreSQL,
інструмент візуалізації PowerBI, бібліотека sclern, machine learning desicion tree;

Постанова Кабінета Міністрів України.

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

1. Інформаційні системи підтримки прийняття рішень
2. Концептуальний підхід до побудови інформаційної системи прийняття рішень при розгляданні заявок на **відновлення** власності громадян України
3. Структура інформаційної системи прийняття рішень та ключові процеси необхідні для її експлуатації

Перелік графічного матеріалу

1._____

2._____

3._____

4._____

Дата видачі завдання_____

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз проблематики сфери | | |
| 2 | Підготування даних | | |
| 3 | Прототипування програми | | |
| 4 | Реалізація системи | | |
| 5 | Висновки+презентація | | |
| 6 | Перевірка роботи на антиплагіат+Передзахист | | |
| 7 | Захист роботи | | |
| 8 | Випуск | | |

Студент _____ О.В. Литвиненко
(підпис) (прізвище та ініціали)

Керівник роботи _____ С.Я. Козаченко
(підпис) (прізвище та ініціали)

Перелік скорочень

СППР(DSS) - Система підтримки прийняття рішень(Decision support system)

RDBMS – (Relational Database Management System) система управління реляційними базами даних

SQL – (Structured Query Language) мова структурованих запитів.

ACID - це абревіатура, яка відображає властивості транзакцій в базі даних: Atomicity(атомарність), Consistency(консистентність), Isolation(ізоляція), Durability (надійність).

IDSS – (Intelligent Decision Support System) інтелектуальна система підтримки прийняття рішень.

AI – (Artificial Intelligence) штучний інтелект.

GPS – (Global Positioning System) глобальна система позиціонування.

CDSS – (Clinical Decision Support System) система підтримки клінічного прийняття рішень

ERP – (Enterprise Resource Planning) планування ресурсів підприємства

ФОП - Фізична Особа-Підприємець

ABC - Abstract Base Classes

ML – (Machine Learning) Машинне навчання

BI – (Business Intelligence) бізнес-інтелект

TIOBE - індекс популярності мов програмування

SSAS – (SQL Server Analysis Services) Сервіс аналізу даних SQL Server

## Вступ

Актуальність теми. В Україні виникла велика криза в вигляді війни з Російською Федерацією. В наслідок військових дій, та бойових зіткнень, а також атак ворога на цивільне населення. Велика кількість людей постраждала, та втратила нерухомість, точніше своє житло. Через обстріли, військові дії, які ведуться безпосередньо на території України, постраждало багато нерухомості, частина населених пунктів знищена. В наслідок цього, велика кількість людей була змушена евакуюватися, в більш безпечні місця, та/або виїхати за кордон. Постало питання, поселення цих людей, збору даних для моделювання перебудови(відбудови) країни, документування пошкоджень. Відновлення того що можна відновити вже зараз. Тобто необхідна компенсація пошкоджень. За наказом Кабінету Міністрів України в Дії була створена послуга “Пошкоджене майно”, де громадяни України можуть подати заяву. Кожна заява повинна бути опрацьована та проаналізована. Так як кількість постраждалих осіб велика, людина фізично не може обробити таку кількість інформації, в короткі строки, та в ручну проаналізувати ці данні. Під час роботи з подібними об’ємами даних, інформація повинна бути нормалізована, перевірена. Бо не можна виключити можливість, помилкових заяв, повторних заяв, через які попередні втрачають актуальність. Звісно кожна заява має бути оброблена, людиною. В той же час для полегшення роботи персоналу над заявами, та більш ефективного використання отриманої інформації, повинна бути впроваджена система підтримки прийняття рішень.

Теоретично-методичні аспекти. Робота систем прийняття рішень не э стандартизованою темою, створено багато класифікацій, типів і компонентів, подібних систем. Але жодна з них не є обов’язковою і далеко не всі системи підпадають під одну з класифікацій. Основними діячами, які працювали з СППР(DSS) є Keen Peter, Sprague R, Power D. J, Haettenschwiler, E. D. Carlson, Holsapple C.W. and A. B. Whinston та інші.

Метою дипломної роботи є створення тестової моделі для перевірки роботи технології, а також попереднього моделювання роботи створеної системи, для подальшої імплементації, в необхідне середовище. А також збільшення ефективності роботи в вище означеній сфері.

Апробація результатів дослідження. Результати дослідження СППР апробовані на II Міжнародній науково-практичній конференції «Системний аналіз, інтелектуальні системи для бізнесу та управління»(23-24 березня 2023р), Київ, ДУТ.

# 1. Інформаційні системи підтримки прийняття рішень

## 1.1. Призначення і види інформаційних систем прийняття рішень

Система підтримки прийняття рішень(Decision support system) (СППР(DSS)) - це інформаційна система, яка допомагає вирішувати бізнесові або організаційні завдання, пов'язані з прийняттям рішень. СППР використовується на керівному, операційному та планувальному рівнях організації (зазвичай на середньому та вищому рівнях управління) та допомагає людям приймати рішення з проблемами, які можуть швидко змінюватися та не бути чітко визначеними наперед - тобто проблемами прийняття неструктурованих та напів структурованих рішень. Системи підтримки прийняття рішень можуть бути повністю автоматизовані, залежно від людини, або комбіновані.

Хоча академіки сприймають СППР як інструмент для підтримки процесів прийняття рішень, користувачі бачать її як інструмент для спрощення організаційних процесів[1]. Деякі автори розширили визначення СППР, щоб включити будь-яку систему, яка може підтримувати процес прийняття рішень, і деякі Системи включають компонент програмного забезпечення для прийняття рішень. Спрейг (1980)[2] визначає правильно названий СППР наступним чином:

СППР зазвичай спрямовані на менш структуровані та недостатньо визначені проблеми, з якими зазвичай стикаються керівники вищого рівня;

СППР намагається поєднати використання моделей або аналітичних методів з традиційним доступом до даних та функціями відновлення даних;

СППР спеціально фокусуються на функціях, які роблять їх легкими у використанні для людей, які не є експертами з комп'ютерів, в інтерактивному режимі;

Та СППР наголошують на гнучкості та пристосованості, щоб врахувати зміни в середовищі та підходах до прийняття рішень користувача.

СППР включають системи на основі знань. Правильно розроблена система - це інтерактивна програмна система, призначена для допомоги приймальникам рішень

компілювати корисну інформацію з комбінації сирих даних, документів та особистих знань або бізнес-моделей, щоб ідентифікувати та вирішувати проблеми та приймати рішення.

Типовою інформацією, яку може зібрати та представити програма прийняття рішень, є наступне:

інвентаризація інформаційних активів (включаючи спадкові та реляційні джерела даних, куби, data warehouses та data marts), порівняльні показники продажів між одним періодом та наступним, прогнозовані показники доходів на основі припущень щодо продажу продукту.

***Компоненти***

Три фундаментальні компоненти архітектури СППР є наступні[3]: база даних (або база знань), модель (тобто контекст прийняття рішень та критерії користувача), інтерфейс користувача. Самі користувачі також є важливими компонентами архітектури.

**База знань**. База знань є невід'ємною частиною бази даних системи прийняття рішень, яка містить інформацію з внутрішніх та зовнішніх джерел. Це бібліотека інформації, що стосується конкретних тем, і є частиною СППР, яка зберігає інформацію, що використовується рушійною системою для визначення шляху дії.

**Система програмного забезпечення**. Система програмного забезпечення складається з систем управління моделями. Модель - це симуляція реальної системи з метою зрозуміти, як працює система та як її можна покращити. Організації використовують моделі, щоб передбачити, як зміниться результат з різними налаштуваннями системи.

Наприклад, моделі можуть бути корисними для розуміння складних систем, які є надто складними, дорогими або небезпечними для повного дослідження у реальному житті. Це ідея комп'ютерних симуляцій, що

використовуються для наукових досліджень, інженерних випробувань, прогнозування погоди та багатьох інших застосувань.

Моделі також можуть використовуватися для представлення та дослідження систем, які ще не існують, наприклад, запропоновані нові технології, планована фабрика або ланцюжок постачання бізнесу. Бізнеси також використовують моделі, щоб передбачити результати різних змін у системі - таких як політики, ризики та регулювання - для допомоги в прийнятті бізнес-рішень.

**Користувацький інтерфейс**. Користувацький інтерфейс забезпечує просту навігацію по системі. Основною метою інтерфейсу користувацької системи підтримки прийняття рішень є забезпечення легкості маніпулювання даними, які зберігаються в ній. Бізнес може використовувати інтерфейс для оцінки ефективності транзакцій DSS для кінцевих користувачів. Користувацькі інтерфейси DSS можуть включати прості вікна, складні інтерфейси з меню та інтерфейси командного рядка.

***Фреймворк розробки(Development frameworks)***

Так само, як і в інших системах, системи підтримки прийняття рішень (DSS) вимагають структурованого підходу[5]. Такий фреймворк включає в себе людей, технології та підхід до розробки. Ранній фреймворк системи підтримки прийняття рішень складається з чотирьох етапів:

*Розвідування* (Intelligence) - пошук умов, які вимагають прийняття рішень;

*Проектування* - розробка та аналіз можливих альтернативних дій або рішень;

*Вибір* - вибір курсу дій серед запропонованих альтернатив;

*Реалізація* - прийняття обраного курсу дій у ситуації прийняття рішення.

Рівні технології DSS (апаратне і програмне забезпечення) можуть включати:

1. Фактичний додаток, який буде використовуватися користувачем. Це частина програми, яка дозволяє приймати рішення в певній проблемній області. Користувач може діяти відповідно до цієї конкретної проблеми.

2. Генератор містить апаратне/програмне забезпечення, яке дозволяє людям легко розробляти конкретні додатки DSS. Цей рівень використовує інструменти врядування або системи, такі як Crystal, Analytica та iThink.
3. Інструменти включають апаратне/програмне забезпечення нижнього рівня. Генератори DSS включають спеціальні мови, функціональні бібліотеки та зв'язуючі модулі.

Ітеративний підхід до розробки дозволяє змінювати та переробляти DSS на різних етапах. Після проектування системи її потрібно протестувати та, якщо потрібно, переробити для досягнення бажаного результату.

***Типи DSS(СППР)***

Використовуючи взаємозв'язок з користувачем як критерій, Haettenschwiler[4] розрізняє пасивні, активні та кооперативні DSS. Пасивна DSS - це система, яка допомагає процесу прийняття рішень, але не може висувати явних рекомендацій або рішень. Активна DSS може висувати такі рекомендації або рішення. Кооперативна DSS дозволяє ітеративний процес між людиною та системою до досягнення узгодженого рішення: приймач рішень (або його радник) може модифікувати, доповнювати або вдосконалювати рекомендації, що надає система, перш ніж відправляти їх назад до системи для перевірки, і так само система знову вдосконалює, доповнює та вдосконалює рекомендації приймача рішень та відправляє їх назад для перевірки.

Ще одна таксономія для DSS, залежно від режиму допомоги, була створена D. Power[3]: він відрізняє комунікаційно-орієнтовану DSS, дані-орієнтовану DSS, документ-орієнтовану DSS, знаннями-орієнтовану DSS та модельно-орієнтовану DSS[3].

1. Комунікаційно-орієнтована DSS сприяє співпраці, підтримуючи роботу кількох людей над спільним завданням; використовує мережу, веб-технології або колаборативні технології, такі як електронна пошта, миттєві повідомлення або голосовий чат, щоб полегшити комунікацію, координацію та участь серед

приймачів рішень, наприклад для управління проектами чи стратегічного планування, щоб дозволити більше одній людині працювати над одним завданням. Метою цього типу DSS є збільшення співпраці між користувачами та системою та покращення загальної ефективності та ефективності системи. Приклади включають інтегровані інструменти, такі як Google Docs або Microsoft SharePoint Workspace.

2. Дані-орієнтована DSS (або DSS на основі даних) зосереджується на доступі до внутрішніх даних компанії та іноді зовнішніх даних, щоб забезпечити інсайти у проблему прийняття рішень. Зазвичай вона використовує бази даних, склади даних або техніки збору даних для виявлення тенденцій та патернів, що дозволяє передбачати майбутні події. Бізнеси часто використовують дані-орієнтовані DSS для дослідження, підсумовування або порівняння даних, щоб допомогти при прийнятті рішень щодо інвентаризації, продажів та інших бізнес-процесів, або для проведення досліджень ринку та оцінки продуктивності. Деякі з них використовуються для прийняття рішень у сфері державного сектору, наприклад, для прогнозування ймовірності майбутньої злочинної діяльності.
3. Документ-орієнтована СПП є типом системи управління інформацією, яка використовує документи для отримання, керування, відновлення та маніпулювання неструктурованою інформацією у різноманітних електронних форматах. Документ-орієнтовані СПП дозволяють користувачам шукати веб-сторінки або бази даних, або знаходити конкретні пошукові запити. Приклади документів, доступних за допомогою документ-орієнтованої СПП, включають політики та процедури, протоколи зустрічей та корпоративні записи.
4. Система підтримки прийняття рішень, що ґрунтується на знаннях (knowledge-driven DSS), забезпечує спеціалізований досвід вирішення проблем, збережений у вигляді фактів, правил, процедур або подібних структур, як інтерактивні дерева рішень та блок-схеми. Knowledge-driven DSS використовує штучний інтелект, експертні системи або техніки машинного навчання для надання порад

або рекомендацій, наприклад, у медичній діагностиці або оцінці кредитоспроможності. У цьому типі системи підтримки прийняття рішень, дані, які використовуються, розміщені в базі знань, яка постійно оновлюється та підтримується системою управління знаннями. Knowledge-driven DSS надає користувачам інформацію, яка відповідає бізнес-процесам та знанням компанії.

5. Модель-орієнтована DSS зосереджена на доступі та маніпулюванні статистичними, фінансовими, оптимізаційними або симуляційними моделями для генерації рішень, наприклад, для розподілу ресурсів або оцінки ризику. Побудовані на основі відповідної моделі прийняття рішень, модель-орієнтовані системи підтримки прийняття рішень використовують дані та параметри, надані користувачами, щоб допомогти приймачам рішень аналізувати ситуацію; вони не обов'язково є даними-інтенсивними. Модель-орієнтовані DSS налаштовуються згідно з попередньо визначеним набором вимог користувача, щоб допомогти аналізувати різні сценарії, що задовольняють ці вимоги. Наприклад, модель-орієнтована DSS може допомогти з плануванням часу або розробкою фінансових звітів, а також може включати моделі оптимізації, симуляції та прогнозування. Dicodess є прикладом генератора відкритого коду модель-орієнтованої DSS.

За критерієм охоплення Power[6] відрізняє системи підтримки прийняття рішень, призначені для всього підприємства (enterprise-wide DSS) та для використання на персональному комп'ютері окремого менеджера (desktop DSS). Системи підтримки рішень для всього підприємства пов'язані з великими складами даних та обслуговують багатьох менеджерів у компанії. Десктоп-система підтримки рішень призначена для роботи на комп'ютері окремого користувача.

**Класифікація**

Існує кілька способів класифікувати додатки DSS. Не кожен DSS гарно вписується в одну з категорій, а може бути комбінацією двох або більше архітектур.

Згідно з Donovan та Madnick (1977)[7], DSS може бути класифікований як інституційний, коли DSS підтримує постійні та повторювані рішення, та як ад-хок, коли DSS підтримує одноразові рішення.

Holsapple та Whinston[8] класифікують DSS на шість категорій: DSS, орієнтованих на текст, DSS, орієнтованих на бази даних, DSS, орієнтованих на електронні таблиці, DSS, орієнтованих на рішення, DSS, орієнтованих на правила, та складний DSS. Остання категорія є найпопулярнішою та є гібридною системою, яка включає дві або більше з п'яти базових структур.

Підтримку, що надається DSS, можна розділити на три різних взаємопов'язаних категорії, визначених Hackathorn та Keen (1981)[9]: персональна підтримка, групова підтримка та організаційна підтримка.

Компоненти DSS можна класифікувати наступним чином:

*Вхідні дан*і: фактори, числа та характеристики для аналізу знання та експертиза користувача: вхідні дані, що вимагають ручного аналізу користувачем

*Вихідні дані*: перетворені дані, на основі яких генеруються "рішення" DSS

*Рішення*: результати, що генеруються DSS на основі критеріїв користувача, які виконують вибрані когнітивні функції прийняття рішень та базуються на технологіях штучного інтелекту або інтелектуальних агентах, називають інтелектуальними системами підтримки прийняття рішень (IDSS)

Нове поле рішення інженерії вважає рішення саме по собі інженерним об'єктом та застосовує принципи інженерії, такі як дизайн та забезпечення якості, до явного представлення елементів, що складають рішення.

***Intelligent decision support system (IDSS)(Інтелектуальні системи прийняття рішень)***

Користувачі також можуть внести штучний інтелект (AI) до систем підтримки прийняття рішень. Ці системи, називаються інтелектуальними системами підтримки прийняття рішень (IDSS), використовують AI для видобування та обробки великих обсягів даних для отримання інсайтів та рекомендацій для поліпшення прийняття рішень. Вони роблять це, аналізуючи різні джерела даних і виявляючи закономірності, тенденції та асоціації для емуляції можливостей людського прийняття рішень.

Створені для дії подібно до людського консультанта, IDSS збирає та аналізує дані для підтримки приймання рішень, ідентифікуючи проблеми, виявляючи та оцінюючи можливі рішення. Компонент AI DSS наближається до можливостей людини, але більш ефективно обробляє та аналізує інформацію як комп'ютерна система.

IDSS може включати розширені можливості, такі як база знань, машинне навчання, видобування даних та інтерфейс користувача. Прикладами реалізації IDSS є гнучкі або розумні виробничі системи, інтелектуальні системи підтримки прийняття рішень в маркетингу та медичні діагностичні системи.

***Додатки***

DSS теоретично може бути створено в будь-якій галузі знань. Один приклад - це система підтримки прийняття рішень для медичної діагностики. Є чотири етапи еволюції системи підтримки клінічних рішень (CDSS): примітивна версія є автономною і не підтримує інтеграцію; друге покоління підтримує інтеграцію з іншими медичними системами; третє засноване на стандартах, а четверте - на моделі сервісу[10].

DSS широко використовується в бізнесі та управлінні. Екзекутивна інформаційна панель та інші програми для визначення продуктивності дозволяють приймати рішення швидше, виявляти негативні тенденції та краще

розподіляти ресурси підприємства. Завдяки DSS, інформація з будь-якої організації представлена у формі графіків, діаграм, тобто у підсумковому вигляді, що допомагає управлінню приймати стратегічні рішення. Наприклад, одним з застосувань DSS є управління та розробка складних систем боротьби з тероризмом[11]. Інші приклади включають перевірку банківським посередником кредитної історії особи, що подає заяву на кредит, або інженерну фірму, що подає заявки на кілька проектів та хоче знати, чи може вона бути конкурентоспроможною за вартістю своїх послуг.

Конкретним прикладом є система канадської національної залізниці, яка регулярно тестує своє обладнання за допомогою системи підтримки прийняття рішень. Одна з проблем з якими стикається будь-яка залізниця - зношені або дефектні рейки, що може призвести до сотень аварій щорічно. За допомогою DSS система канадської національної залізниці змогла зменшити кількість аварій, тоді як інші компанії зафіксували збільшення їхньої кількості..

DSS використовувалися для оцінки ризиків та інтерпретації моніторингових даних з великих інженерних споруд, таких як дамби, вежі, кафедральних соборів, чи будівлі з цегли. Наприклад, Mistral є експертною системою для моніторингу безпеки дамб, розробленою в 1990-х роках компанією Ismes (Італія). Вона отримує дані від автоматичної системи моніторингу та здійснює діагностику стану дамби. Її перший екземпляр, встановлений в 1992 році на дамбі Рідраколі (Італія), все ще працює цілодобово 24/7/365[12]. Вона була встановлена на кількох дамбах в Італії та за кордоном (наприклад, на дамбі Ітаїпу в Бразилії)[13] та на пам'ятниках під назвою Kaleidos[14]. Mistral є зареєстрованою торговою маркою CESI. GIS успішно використовували з 90-х років разом з DSS, щоб показати на мапі оцінки ризиків в реальному часі, засновані на даних моніторингу, зібраних в районі катастрофи Вал Пола (Італія)[15].

*Приклади DSS*

Організації використовують системи підтримки прийняття рішень в декількох різних контекстах, зокрема:

**GPS routing(**Маршрутизація за допомогою GPS**)**. Планування маршруту з використанням GPS є типовим прикладом системи підтримки прийняття рішень. Вона порівнює різні маршрути, враховуючи фактори, такі як відстань, час та вартість поїздки. Система навігації GPS також дозволяє користувачам вибирати альтернативні маршрути, відображаючи їх на мапі та надаючи інструкції крок за кроком.

**ERP dashboards (**Дашборди ERP**)**. Дошка із планування ресурсів підприємства (ERP, enterprise resource planning) може використовувати систему підтримки прийняття рішень для візуалізації змін у виробничих та бізнес-процесах, моніторингу поточної бізнес-продуктивності в порівнянні з встановленими цілями та виявлення областей для поліпшення. Дошки ERP дозволяють власникам бізнесу бачити знімок найважливіших чисел та метрик їх компанії.

**Clinical decision support system(**Система підтримки прийняття рішень в медицині**).** Система підтримки прийняття рішень в медицині (CDSS) - це програмне забезпечення, яке використовує розроблені алгоритми для допомоги лікарям у прийнятті найкращих медичних рішень. Медичні фахівці часто використовують їх для інтерпретації медичних записів і результатів тестування, а також для розрахунку найкращого плану лікування. CDSS у галузі охорони здоров'я можуть допомогти провайдерам виявляти відхилення під час певних тестів, а також контролювати стан пацієнтів після певних процедур для визначення наявності будь-яких небажаних реакцій.

## 1.2. Підходи та критерії вибору систем прийняття рішень

*Переваги Систем підтримки прийняття рішень(DSS)*

Система підтримки прийняття рішень збільшує швидкість та ефективність процесу прийняття рішень, оскільки може збирати та аналізувати дані в режимі реального часу.

Вона сприяє навчанню всередині організації, оскільки для впровадження та функціонування системи підтримки прийняття рішень необхідно розвивати певні навички.

Вона автоматизує монотонні менеджерські процеси, що означає, що більше часу менеджера може бути витрачено на процес прийняття рішень.

Вона покращує міжособистісну комунікацію всередині організації.

*Недоліки систем підтримки прийняття рішень (Decision Support System)*

Витрати на розробку та впровадження DSS є значним капіталовкладенням, що робить його менш доступним для менших організацій.

Компанія може стати залежною від DSS, оскільки він інтегрований у повсякденні процеси прийняття рішень для поліпшення ефективності та швидкості. Однак, керівники зазвичай занадто покладаються на систему, що позбавляє прийняття рішень суб’єктивної складової.

DSS може призвести до перенасичення інформацією, оскільки інформаційна система має тенденцію враховувати всі аспекти проблеми. Це створює дилему для кінцевих користувачів, оскільки вони залишаються з кількома варіантами вибору.

Впровадження DSS може викликати страх та негативну реакцію серед працівників нижчого рівня. Багато з них не впевнені у новій технології та бояться втратити свою роботу через технології.

*DSS характеристика*

Коли розглядається придатність та ефективність DSS для розв'язання проблеми прийняття рішень, потрібно враховувати інші характеристики, крім типу DSS. Важливими факторами є взаємодія, адаптивність, інтелектуальність та простота

використання. Взаємодія означає ступінь, до якого приймач рішень може взаємодіяти з DSS та контролювати вхідні, вихідні данні і процеси прийняття рішень. Адаптивність - це ступінь, до якого DSS може пристосовуватися до змінних потреб, вподобань та відгуків приймача рішень. Інтелектуальність - це ступінь, до якого DSS може виконувати складні, складні та автономні завдання, що імітують людський інтелект. Нарешті, простота використання - це ступінь, до якого DSS легко використовувати, вивчати та зрозуміти приймачем рішень. Високий рівень кожної з цих характеристик може покращити ефективність DSS.

*Вибір DSS*

Вибір найкращого типу та характеристик DSS для задачі прийняття рішень - не просте завдання. Це вимагає ретельного аналізу контексту, об'єктивів, критеріїв та обмежень прийняття рішень, а також наявності, якості та сумісності даних, моделей, знань та ресурсів зв'язку. Щоб допомогти у процесі вибору DSS, спочатку слід визначити задачу прийняття рішень та її масштаб, складність та терміновість. Визначте приймачів рішень та їхні ролі, обов'язки та переваги. Визначте бажані результати та показники продуктивності процесу прийняття рішень та DSS. Далі перегляньте наявні типи та характеристики DSS та оцініть їх переваги та недоліки для задачі прийняття рішень. Порівняйте та контрастуйте особливості, функції та переваги різних варіантів DSS. Розгляньте компроміси та ризики, пов'язані з кожною опцією. Нарешті, виберіть найбільш підходящий тип та характеристики DSS, які відповідають задачі прийняття рішень та приймачам рішень. Обґрунтуйте вибір на основі аналізу та оцінки варіантів DSS. Документуйте критерії вибору, припущення та обґрунтування. Впроваджуйте та тестуйте обраний DSS та контролюйте його продуктивність та вплив на задачу прийняття рішень та приймачів рішень. Збирайте відгуки та дані про використання, результати та задоволеність DSS. Виявляйте та вирішуйте будь-які питання або проблеми, які виникають під час впровадження та функціонування системи.

Вибір правильної системи підтримки прийняття рішень (DSS) є критичним рішенням, яке вимагає ретельного розгляду. Ось деякі критерії та способи вибору DSS:

1. Визначте проблему прийняття рішень: першим кроком у виборі DSS є чітке визначення проблеми прийняття рішень. Це допоможе визначити тип DSS, необхідний для вирішення проблеми та функції, необхідні для її вирішення.
2. Врахуйте вимоги користувачів: необхідно враховувати вимоги користувачів під час вибору DSS. DSS повинна бути користувацькій зрозумілою та мати функції, які відповідають потребам приймачів рішень.
3. Оцініть вимоги до даних: DSS повинна бути здатна впоратися з вимогами до даних проблеми прийняття рішень. Це включає обсяг даних, якість даних та тип даних, необхідних для аналізу.
4. Оцініть процес прийняття рішень: DSS повинна добре вписуватися у процес прийняття рішень. Вона повинна підтримувати процес прийняття рішень та забезпечувати своєчасну та точну інформацію для приймачів рішень.
5. Оцініть технічні вимоги: DSS повинна бути сумісна з існуючими апаратними та програмними системами в організації. Вона також повинна бути масштабованою та гнучкою для врахування майбутнього зростання.
6. Розгляньте витрати: Вартість DSS повинна бути відповідною до користі, яку вона надає. Вона включає в себе вартість придбання, щоденні технічні роботи та витрати на підтримку.
7. Розгляньте вимоги до безпеки та конфіденційності: DSS повинна мати відповідні можливості забезпечення безпеки та конфіденційності для захисту конфіденційної інформації.
8. Розгляньте вимоги до навчання та підтримки: DSS повинна мати відповідну навчання та підтримку, щоб користувачі могли ефективно використовувати систему.

Способи вибору DSS:

Послухайте експертів: Порадьтеся з експертами у своїй галузі, щоб визначити потрібний тип DSS для розв'язання проблеми прийняття рішень.

Проведіть дослідження: Проведіть дослідження доступних варіантів DSS та порівняйте їх можливості та функції.

Запросіть пропозиції: Запросіть пропозиції від постачальників, які надають рішення на основі DSS, та оцініть їх на основі вищезазначених критеріїв.

Проведіть тестування: Перевірте DSS, щоб переконатися, що вона відповідає вимогам проблеми прийняття рішень та організації.

Вибір правильної DSS є ключовим чинником успіху процесу прийняття рішень. Враховуючи критерії та способи вибору DSS, організації можуть визначити найкращу систему для підтримки потреб прийняття рішень.

## 2. Концептуальний підхід до побудови інформаційної системи прийняття рішень при розгляданні заявок на відновлення власності громадян України

### 2.1. Актуальність і специфіка сфери використання системи прийняття рішень при розгляданні заявок на відновлення власності громадян України

Актуальність використання системи прийняття рішень у розгляді заявок на відшкодування збитків і відновлення майна громадян України полягає в тому, що такі заявки можуть бути досить складними і містити велику кількість інформації, що потребує аналізу та обробки.

У процесі розгляду заявок на відшкодування збитків і відновлення майна громадянам, відповідальні посадовці повинні приймати обґрунтовані рішення, що базуються на об'єктивних фактах та відповідному аналізі інформації. Використання системи прийняття рішень дозволяє підвищити ефективність роботи заявників, прискорити процес розгляду заявок та знизити ймовірність прийняття неправильного рішення.

Особливості сфери відшкодування збитків і відновлення майна полягають у необхідності аналізувати багато різних факторів, що впливають на рішення. Наприклад, необхідно аналізувати ступінь пошкодження майна, обсяг збитків, тощо.

Система прийняття рішень може використовуватися для автоматичного аналізу даних та інформації, що знижує кількість помилок і забезпечує більш точну та об'єктивну оцінку ризиків та наслідків. Використання системи прийняття рішень також забезпечує стандартизацію процесу розгляду заявок та відновлення майна громадян, що дозволяє підвищити рівень його якості та ефективність.

В попередньому році в наслідок військової агресії Російської федерації, в країні з’явилася проблема пошкодження нерухомості, особливо в прифронтових зонах. Населення було вимушене змінити місце проживання, на тимчасові прихистки. З’явилися проблема відшкодування, відновлення, та проживання переміщених осіб. За для вирішення даної проблеми Кабінет Міністрів України Постановив від 26

березня 2022 р. № 380 «Про збір, обробку та облік інформації про пошкоджене та знищене нерухоме майно внаслідок бойових дій, терористичних актів, диверсій, спричинених військовою агресією Російської Федерації»[15]

Ця постанова дає можливість збору та обробки даних про пошкодження через застосунок ДІЯ. Кожен громадянин може подати заяву про пошкодження житла та в майбутньому отримати компенсацію чи необхідне відновлення.

Таким чином створюється база даних, з інформацією про пошкодження та кількість постраждалих.

Нажаль, об'єм такої базі даних, доволі великий і уся надана інформація має бути оброблена, проаналізована та нормалізована. Помилкові данні повинні бути ідентифіковані і відділені від основної частини, де з годом мають також бути опрацьовані.

Надалі потрібно розглянути що таке застосунок ДІЯ і як працює подання заявок і з чого вона складається.

**Дія** (скорочення від **«Держава і я»**) — Міністерство цифрової трансформації України розробило мобільний додаток, веб-портал і бренд для цифрової держави в Україні[16]. Презентація цієї ініціативи відбулася у 2019 році, а її офіційний запуск відбувся у 2020 році[17]. Застосунок дозволяє користувачам зберігати свої водійські посвідчення, внутрішні та закордонні паспорти та інші документи в їх смартфонах. Крім того, користувачі можуть передавати копії своїх документів при отриманні банківських та поштових послуг, заселенні в готелях та інших життєвих ситуаціях. Через Дію (застосунок і/або портал) також можна отримати державні послуги, такі як єМалятко (комплексна послуга при народженні дитини), зареєструвати бізнес та ФОП онлайн, сплачувати податки та подавати декларації, підписувати будь-які документи, змінювати місце реєстрації та багато іншого. До 2024 року планується перевести усі державні послуги до Дії. Станом на травень 2022 року,

застосунком і порталом користується понад 17 мільйонів людей, доступно 72 послуги на порталі і 9 послуг та 15 цифрових документів у застосунку[18][19].

Перейдемо до послуги яка нас цікавить.

Опис послуги:

Повідомте про пошкодження або знищення своєї нерухомості внаслідок військової агресії Російської Федерації. Згодом буде створено спеціальну комісію, яка оцінить збитки. Про порядок та процедуру компенсації буде повідомлено пізніше. Пошкодження побутової техніки чи ремонту вказувати не потрібно.

Онлайн

1. Зареєструйтеся або авторизуйтеся на порталі в кабінеті громадянина на diia.gov.ua за допомогою електронного підпису чи BankID.

2. Заповніть онлайн-форму та вкажіть:- дані про об’єкт, що було пошкоджено (у разі, якщо інформація про ваш об’єкт зберігається в Державному реєстрі речових прав, її може бути завантажено автоматично. В іншому разі внесіть інформацію про об’єкт самостійно;- фото та опис пошкоджень

3. Ознайомтесь з повідомленням та надішліть його.

Наступні кроки

Пізніше буде створена спеціальна комісія, яка оцінить збитки та визначить порядок та процедуру компенсації. Компенсація буде виплачуватись за повністю зруйновані або частково пошкоджені житлові будинки, квартири та житлові приміщення, які є власністю громадян України. Наразі компенсації за нежитлову нерухомість не передбачені. Механізм нарахування компенсації ще не визначений, але найімовірніше, потрібно буде подати окрему заявку на оцінку та виплату компенсації, після чого спеціальна комісія визначить суму виплати. У деяких випадках можливі інші види компенсації, наприклад, видача нової квартири замість знищеної. Компенсації наразі не передбачені за пошкоджені транспортні засоби та нежитлові приміщення.

Які є поля в цій послузі.

Перший, це тип нерухомого майна.

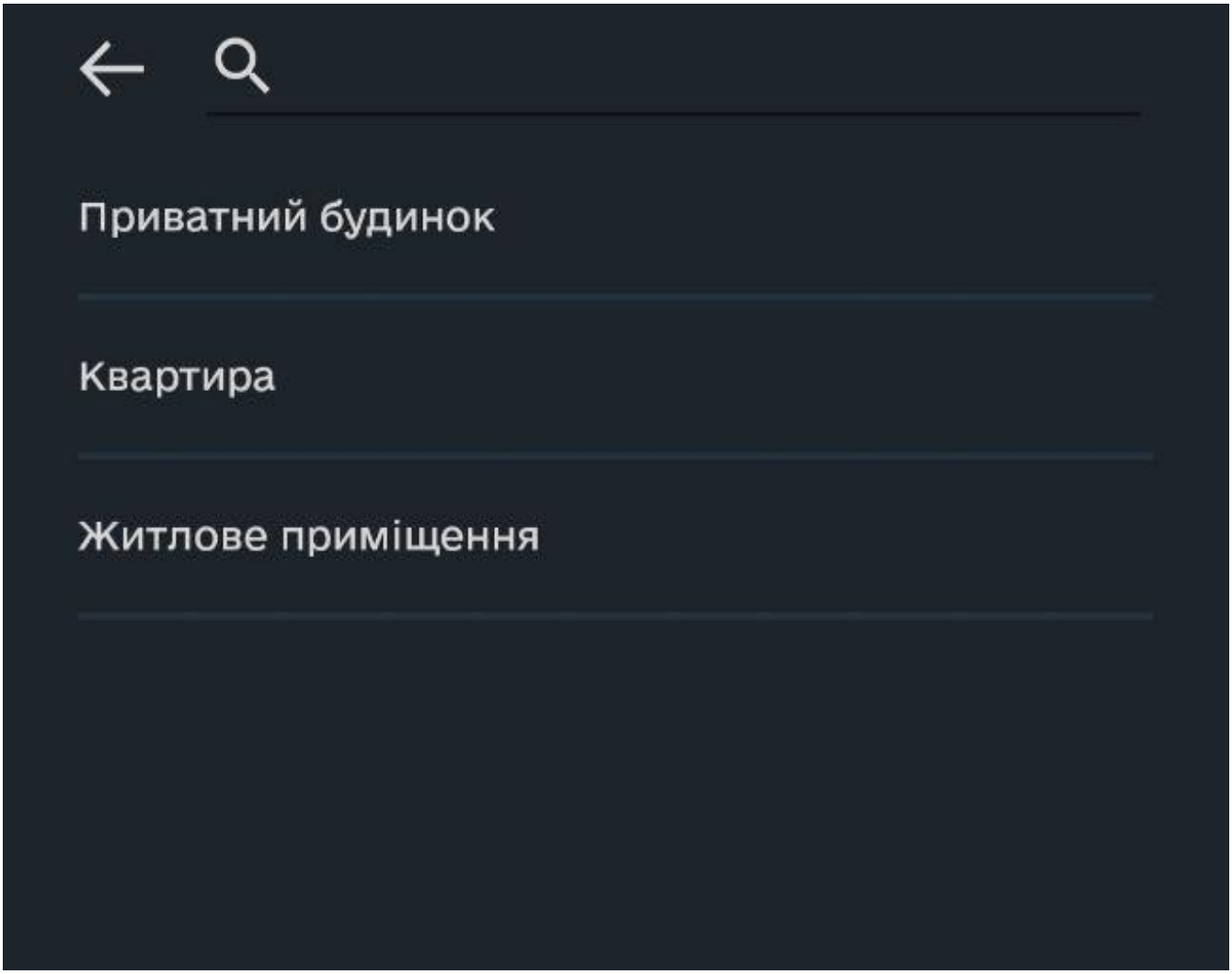



Рис. 2.1.1 Тип нерухомого майна в застосунку ДІЯ.

Приватні будинки, квартири, та житлові приміщення. В моделі це поле є одним з головних параметрів, за допомогою яких приймається рішення. В законодавстві, чітко встановлені розмір цих житлових приміщень, а також максимальна кількість людей яка може проживати в цих типах нерухомості. Житлове приміщення це виняток. Згідно з законодавством України, житлове приміщення - це будь-який окремий об’єкт нерухомого майна, який призначений для постійного проживання людини або групи людей. Відповідно

до цього визначення, житловим приміщенням може бути квартира в багатоквартирному будинку, окремий будинок, кімната в гуртожитку, апартаменти в готелі або будь-який інший об’єкт нерухомого майна, призначений для постійного проживання. Тому в моделі це автоматично буде відмічена як rejected, бо кожний випадок потребує участі людини, а не комп’ютерної моделі.

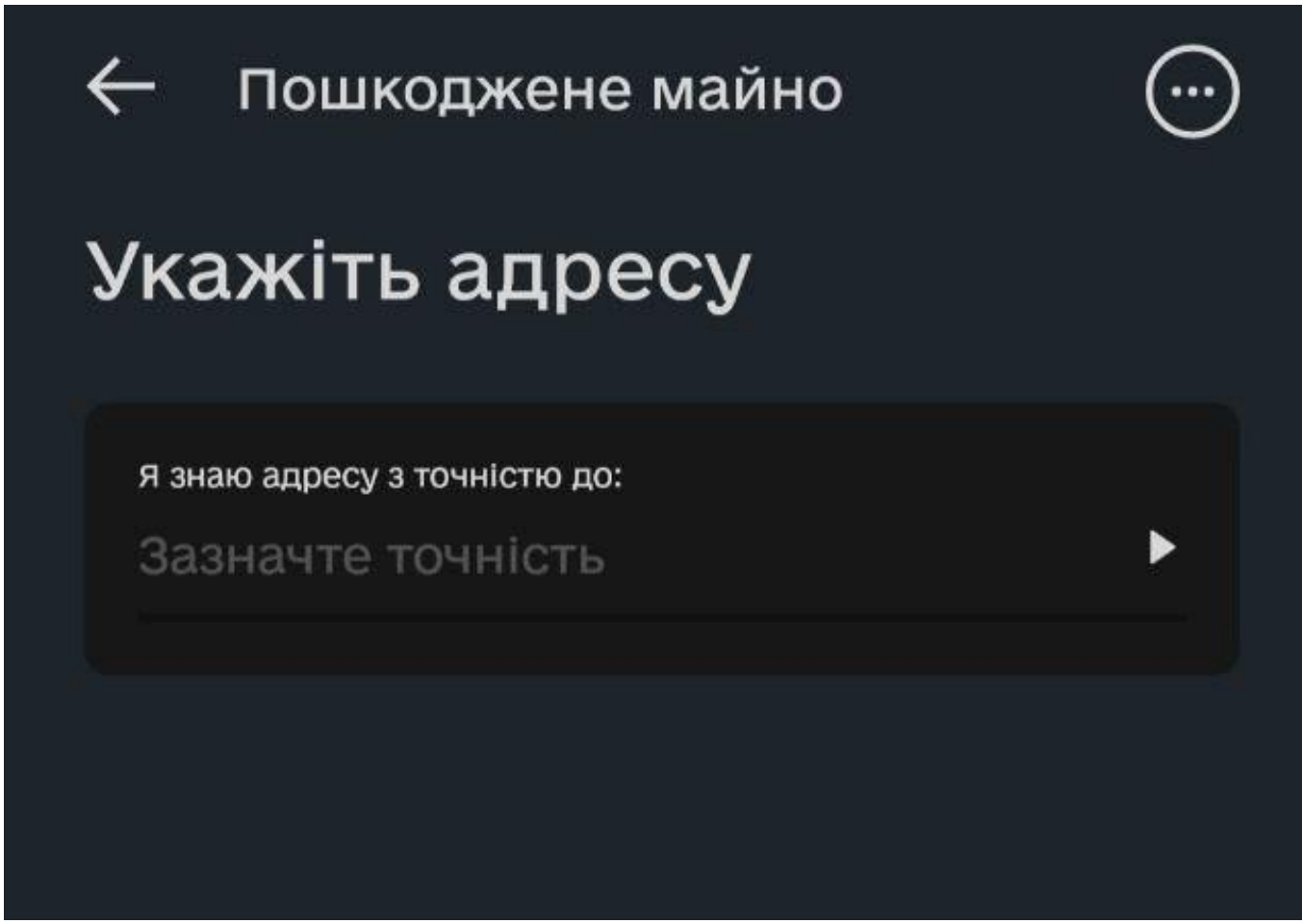



Рис. 2.1.2 Адреса в застосунку ДІЯ.

Фактично в моделі не приймає участі. Адреса, має велике значення при розробці моделі яка буде підключена до ДІЇ, за для того, щоб ідентифікувати майно, його існування, та місцезнаходження.

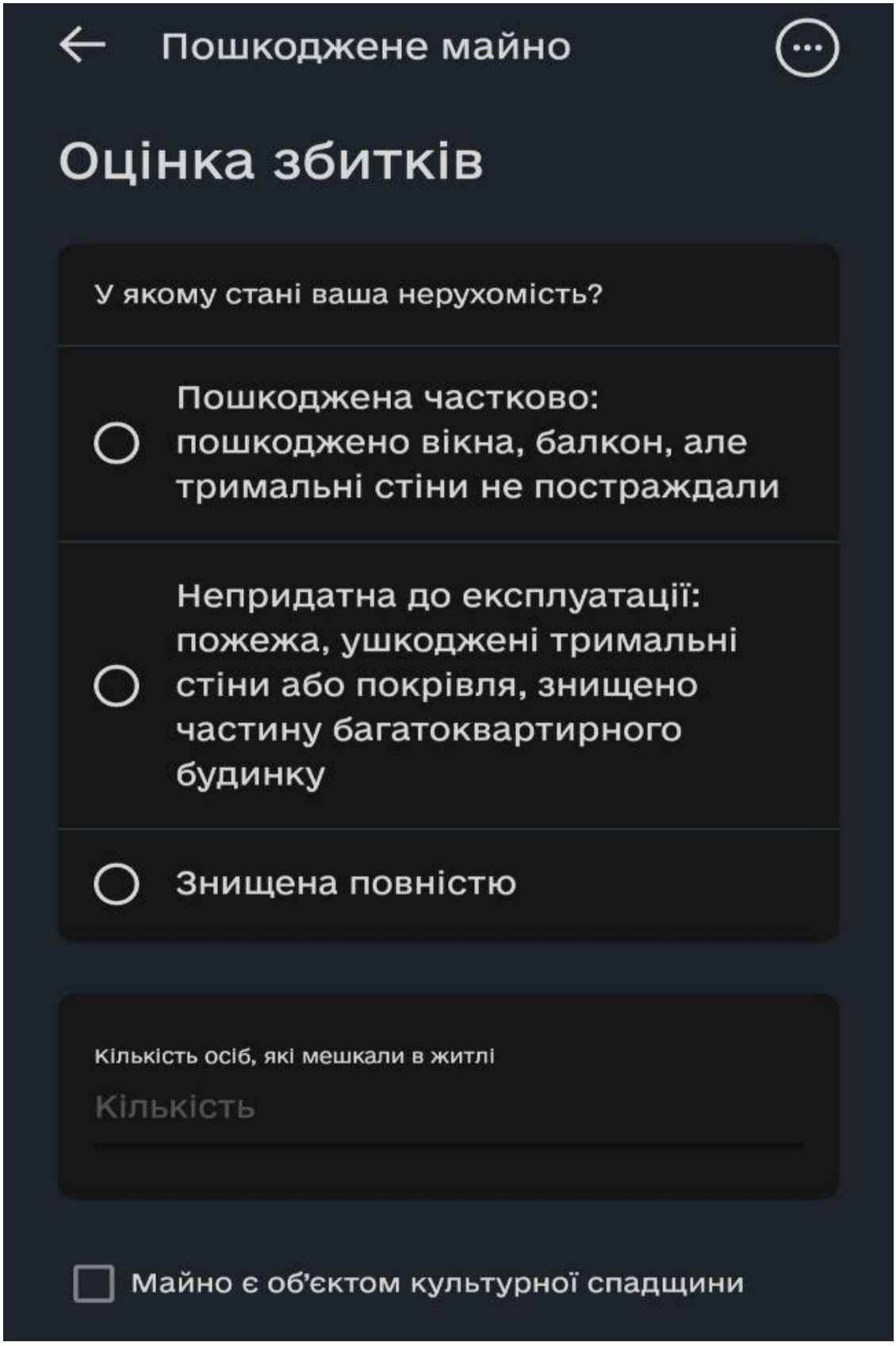



Рис. 2.1.3 Оцінка збитків в застосунку ДІЯ.

Оцінка збитків, розширює модель та вносить класифікацію, в якій буде ступінь важливості, і складності по відновленню та компенсації. Але в моїй моделі його зараз немає.

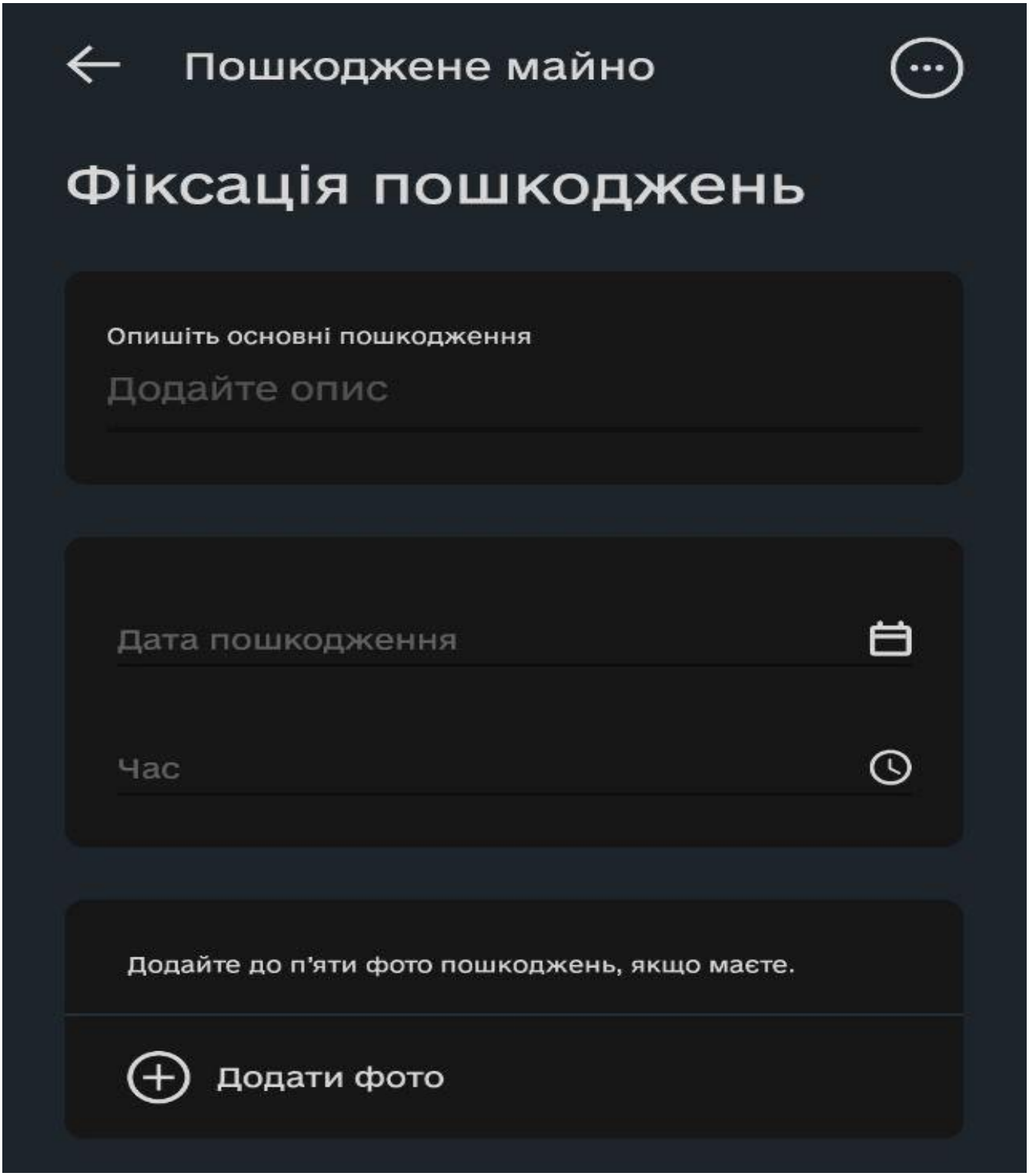



Рис. 2.1.4 Фіксація пошкоджень застосунку ДІЯ.

Фіксація пошкоджень, на даний момент не приймає участі в моделі так як воно спрямовано на підтвердження існування пошкодження, а також для роботи оціночної комісії, але не є обов’язковим, через неможливість подати фотографічні докази частини пошкоджень через знаходження власності на окупованій території чи в зоні бойових дій.

## 2.2. **Вибір и обґрунтування підходові до побудови інформаційної системи прийняття рішень при розгляданні заявок на відновлення власності громадян України**

### *Python*

Як основа для написання моделі була використана мова програмування Python, бо вона має багато переваг перед іншими мовами програмування в аспекті роботі з даними.

Python є інтерпретованою, об'єктно-орієнтованою мовою програмування високого рівня з динамічною семантикою. Її вбудовані структури даних високого рівня, поєднані з динамічним набором типів та динамічним зв'язуванням, роблять її дуже привабливою для швидкої розробки програм, а також для використання як мови скриптів або мови з'єднання, щоб з'єднати існуючі компоненти разом. Простий та легкий у вивченні синтаксис Python підкреслює зрозумілість коду та зменшує вартість підтримки програм. Python підтримує модулі та пакети, що сприяє модульності програм та повторному використанню коду. Інтерпретатор Python та велика стандартна бібліотека доступні у вихідному або бінарному вигляді безкоштовно для всіх основних платформ та можуть бути вільно поширені.

Гвідо ван Россум почав працювати над мовою програмування Python наприкінці 1980-х років як наступник мови програмування ABC і випустив першу версію Python 0.9.0[20] в 1991 році. Python 2.0 був випущений у 2000 році. Python 3.0, який був випущений у 2008 році, був значним оновленням, яке

не було повністю сумісним з попередніми версіями. Python 2.7.18, випущений у 2020 році, був останнім релізом Python 2[21].

Python - це багато парадигмова мова програмування. Об'єктно-орієнтоване програмування та структуроване програмування повністю підтримуються, і багато з їх функцій підтримують функціональне програмування та аспектно-орієнтоване програмування (включаючи мета програмування[22] та мета об’єкти). Багато інших парадигм підтримуються за допомогою розширень, включаючи дизайн за контрактом та логічне програмування[23].

Python використовує динамічний типізм та поєднання рахунку посилань та автоматичного збирача сміття з циклом виявлення для управління пам'яттю. Для прив'язки імен використовується динамічне розв'язування (пізне зв'язування), яке пов'язує імена методів та змінних під час виконання програми.

В дизайні мови передбачено деяку підтримку функціонального програмування в традиції Lisp. Мова має функції filter, map і reduce; comprehensions для списків, словників, множин і генераторів виразів. В стандартній бібліотеці є два модулі (itertools та functools), які реалізують функціональні інструменти, запозичені з Haskell та Standard ML.

Основна філософія мови Python узагальнена в документі "The Zen of Python" (PEP 20) і включає афоризми, такі як[24]:

Красиве краще за потворне.

Явне краще за неявне.

Просте краще за складне.

Складне краще за заплутане.

Читабельність має значення..

Замість включення всієї функціональності безпосередньо до ядра мови, Python був розроблений як високоякісно розширювана завдяки модулям. Ця компактна модульність зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих додатків. Бачення Ван Россума про невелику мову ядра з

великою стандартною бібліотекою та легко розширюваним інтерпретатором виникла з його роздратування на ABC, який прихильно сприймав протилежний підхід[25].

Python намагається мати простіший, менш заплутаний синтаксис та граматику, дозволяючи розробникам вибирати свою методологію програмування. На відміну від гасла Perl "є більше одного способу це зробити", Python прихильний до філософії "повинен бути один - і краще всього тільки один - очевидний спосіб це зробити". Алекс Мартеллі, член Фонду програмного забезпечення Python та автор книг про Python, написав: "Описуючи щось як 'хитре', в культурі Python це не вважається компліментом"[26].

Розробники Python прагнуть уникнути передчасної оптимізації та відкидають заплатки до не критичних частин CPython еталонної реалізації, які можуть забезпечити незначні підвищення швидкості за рахунок збереження чіткості. Якщо швидкість є важливою, програміст Python може перемістити функції, що вимагають часу, до модулів розширення, написаних на мовах, таких як C; або використовувати PyPy, компілятор з миттєвою компіляцією. Також доступний Cython, який перекладає скрипт Python у C і виконує прямі виклики API на рівні C до інтерпретатора Python.

У спільноті Python поширене новотвірне слово "pythonic", яке має широкий спектр значень, пов'язаних зі стилем програмування. "Pythonic" код може добре використовувати ідіоми Python, бути природним або відображати вміння використовувати мову програмування, або відповідати мінімалістичній філософії Python і акцентувати на читабельності. Код, який складно зрозуміти або виглядає як грубий переклад з іншої мови програмування, називається "unpythonic".

З 2003 року Python стабільно потрапляє до десятки найпопулярніших мов програмування за версією індексу спільноти програмістів TIOBE, де на грудень 2022 року став найпопулярнішою мовою програмування (перед C, C++ та Java).

У 2007, 2010, 2018 та 2020 роках вона була обрана мовою програмування року за "найбільше зростання рейтингу за рік" (єдина мова, яка це зробила чотири рази на 2020 рік).

Одне емпіричне дослідження встановило, що скриптові мови, такі як Python, є більш продуктивними, ніж традиційні мови, такі як C та Java, для задач програмування, пов'язаних з маніпулюванням рядків та пошуком у словнику, і визначили, що використання пам'яті часто "краще, ніж у Java, і не набагато гірше, ніж у C або C ++".

Великі організації, що використовують Python, включають Wikipedia, Google, Yahoo !, CERN, NASA, Facebook, Amazon, Instagram, Spotify, а також деякі менші суб'єкти, такі як ILM і ITA. Соціальний новинний сайт Reddit був написаний переважно на Python.

### *Decision tree*

Рушійною силою моделі, та її основою є алгоритм машинного навчання(machine learning) decision tree(дерево рішення). Дерево рішень - це ієрархічна модель прийняття рішень, що використовує деревоподібну структуру рішень та їх можливих наслідків, включаючи випадкові наслідки, ресурсні витрати та корисність. Це один з способів відображення алгоритму, який містить лише умовні оператори контролю.

Дерева рішень часто використовуються в операційному дослідженні, зокрема в аналізі прийняття рішень[27], щоб допомогти визначити стратегію, яка найбільш імовірно досягне мети, а також є популярним інструментом у машинному навчанні.

Дерево рішень - це структура, схожа на блок-схему, в якій кожен внутрішній вузол представляє "тест" на атрибуті (наприклад, чи випав грош або орел), кожна гілка представляє результат тесту, а кожний кінцевий вузол представляє мітку класу (рішення, прийняте після обчислення всіх атрибутів). Шляхи від кореня до кінцевих вузлів представляють правила класифікації.

У прийнятті рішень дерево рішень та тісно пов'язана діаграма впливу використовуються як візуальний та аналітичний інструмент прийняття рішень, де

розраховуються очікувані значення (або очікувана корисність) конкуруючих альтернатив.

Дерево рішень складається з трьох типів вузлів[28]:

**Вузли рішень** - зазвичай представлені квадратами

**Вузли ймовірності** - зазвичай представлені кілочками

**Кінцеві вузли** - зазвичай представлені трикутниками

Дерева рішень часто використовуються в дослідженнях операцій та управлінні операціями. Якщо на практиці потрібно приймати рішення в режимі реального часу при неповній інформації, то дерево рішень повинно бути поєднане з моделлю ймовірності як кращої моделі вибору або алгоритму вибору в режимі онлайн. Ще одним застосуванням дерев рішень є описові засоби для розрахунку умовних ймовірностей.

Дерева рішень, діаграми впливу, функції корисності та інші інструменти та методи аналізу рішень викладаються для студентів бакалаврської програми в школах бізнесу, здоров'я та громадського здоров'я і є прикладами методів досліджень операцій або наук про управління.

Серед інструментів підтримки прийняття рішень, дерева рішень (та діаграми впливу) мають кілька переваг. Дерева рішень:

Легкі у розумінні та інтерпретації. Після короткого пояснення люди здатні зрозуміти моделі дерев рішень.

Мають цінність навіть при малій кількості жорстких даних. Важливі уявлення можуть бути згенеровані на основі експертного опису ситуації (її альтернатив, ймовірностей та витрат) та їх вподобань до результатів.

Допомагають визначати найгірший, найкращий та очікуваний результати для різних сценаріїв.

Використовують модель білого ящика. Якщо заданий результат надається моделлю.

Можуть бути поєднані з іншими техніками прийняття рішень.

Може бути врахована діяльність більше, ніж одного приймача рішень.

Недоліки decision trees:

Вони є нестійкими, що означає, що невелика зміна в даних може призвести до значної зміни структури оптимального дерева рішень.

Вони часто є відносно неточними. Багато інших передбачувачів працюють краще зі схожими даними. Це може бути виправлено, замінивши одне дерево рішень на випадковий ліс дерев рішень, але випадковий ліс не так легко інтерпретувати, як одне дерево рішень.

Для даних, що включають категоріальні змінні з різною кількістю рівнів, інформаційний приріст в деревах рішень сприяє тим атрибутам, які мають більше рівнів.

Обчислення можуть стати дуже складними, особливо якщо багато значень є невизначеними і / або якщо багато результатів пов'язані між собою.

Простий приклад Decision tree

У цьому прикладі людина намагатиметься вирішити, чи варто їй піти на комедійне шоу чи ні.

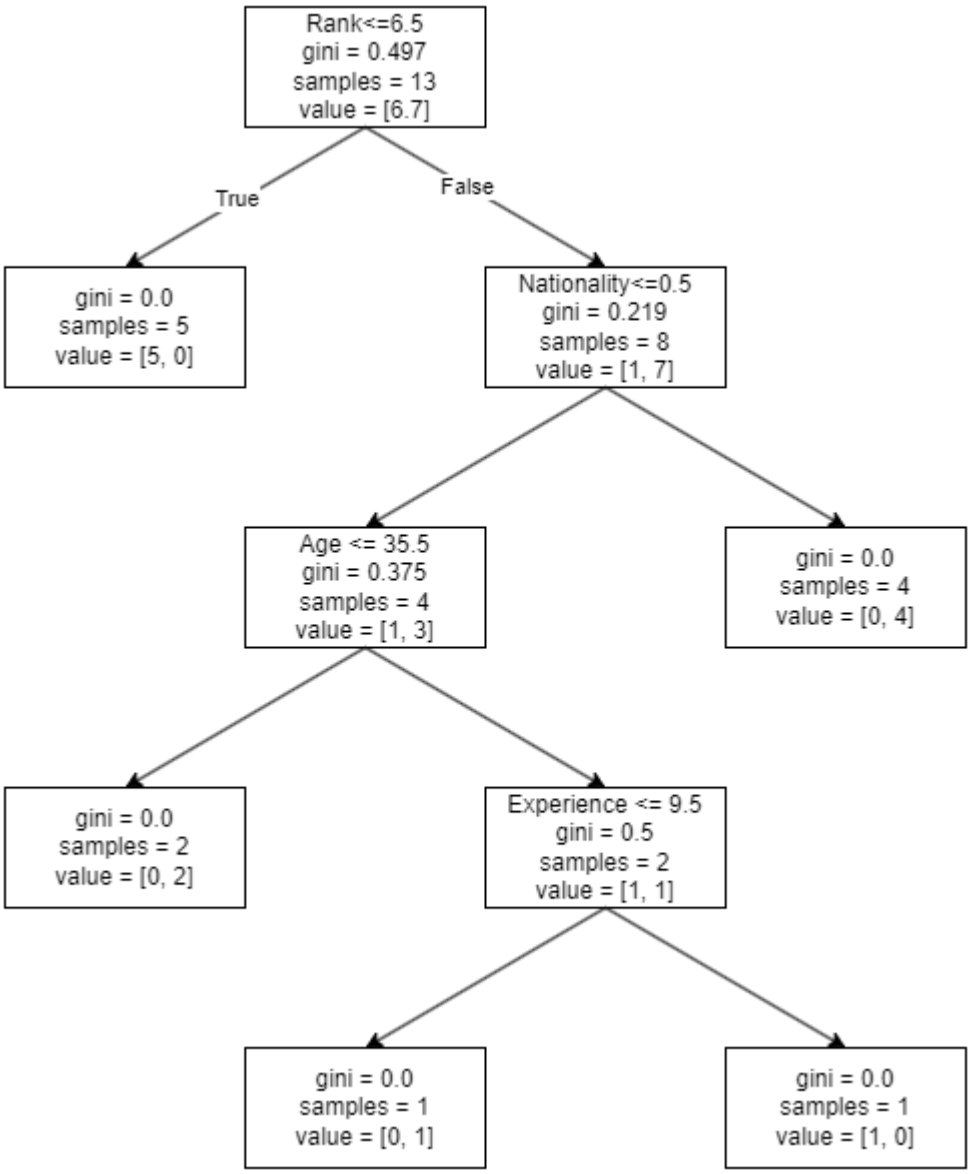



Рис. 2.2.1 Схематичний приклад роботи Decision tree.

Таблиця. 2.2.1 Данні для прикладу Decision tree.

| Age | Experience | Rank | Nationality | Go |
|---|---|---|---|---|
| 36 | 10 | 9 | UK | NO |
| 42 | 12 | 4 | USA | NO |
| 23 | 4 | 6 | N | NO |
| 52 | 4 | 4 | USA | NO |
| 43 | 21 | 8 | USA | YES |

| | | | | |
|---|---|---|---|---|
| 44 | 14 | 5 | UK | NO |
| 66 | 3 | 7 | N | YES |
| 35 | 14 | 9 | UK | YES |
| 52 | 13 | 7 | N | YES |
| 35 | 5 | 9 | N | YES |
| 24 | 3 | 5 | USA | NO |
| 18 | 3 | 7 | UK | YES |
| 45 | 9 | 9 | UK | YES |

*Пояснення результатів*

Дерево рішень використовує ваші попередні рішення, щоб розрахувати ймовірність того, чи ви захочете піти на виступ коміка чи ні.

Давайте розглянемо різні аспекти дерева рішень:

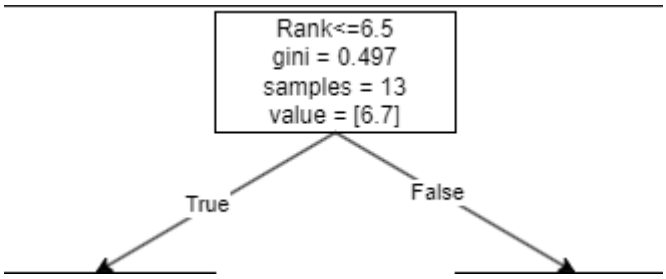



Рис. 2.2.2 Схематичний приклад роботи Decision tree. Частина 1.

Rank

Rank <= 6.5  означає, що кожен комік з рейтингом 6.5 або нижче піде за True стрілку (наліво), а решта піде за False стрілку (направо).

gini = 0.497  відноситься до якості розбиття і завжди є числом від 0,0 до 0,5, де 0,0 означає, що всі зразки отримали один і той же результат, а 0,5 означає, що розбиття зроблено саме по середині.

samples =  13 означає, що на цій стадії рішення залишилося 13 коміків, що є всіма, оскільки це перший крок.

value = [6, 7] означає, що з цих 13 коміків, 6 отримають відповідь "NO", а 7 - "GO".

Gini

Існує багато способів розбити вибірку на підвибірки, в цьому навчальному посібнику ми використовуємо метод Gini. Метод Gini використовує таку формулу: Gini = 1 - (x/n)2 - (y/n)2 Де x - це кількість позитивних відповідей ("GO"), n - кількість вибірок, а y - це кількість негативних відповідей ("NO"), що дає нам наступний розрахунок: 1 - (7 / 13)2 - (6 / 13)2 = 0.497

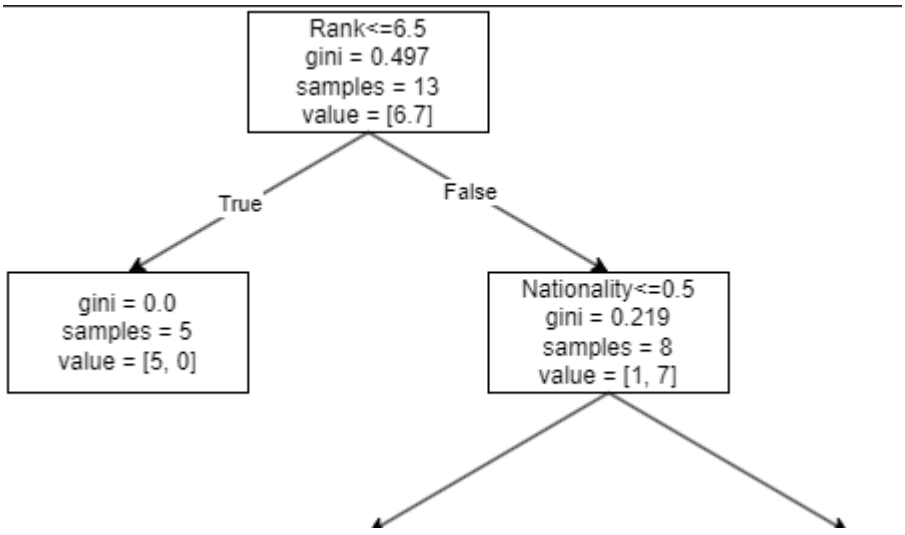



Рис. 2.2.3 Схематичний приклад роботи Decision tree. Частина 2.

Наступний крок містить дві скриньки: одну для коміків з рейтингом 6,5 або менше і іншу для решти.

True - 5 коміків закінчуються тут:

gini = 0.0 означає, що всі зразки отримали одинаковий результат.

samples = 5 означає, що залишилося 5 коміків в цій гілці (5 коміків з рейтингом 6,5 або менше).

value = [5, 0] означає, що 5 отримають " NO ", а 0 отримають " GO ".

False - 8 коміків продовжують:

Nationality

Nationality <= 0.5 означає, що коміки з національністю менше 0,5 підуть стрілкою вліво (що означає всіх з Великої Британії), а решта підуть стрілкою вправо.

gini = 0.219 означає, що близько 22% зразків будуть іти в одному напрямку.

samples = 8 означає, що в цій гілці залишилося 8 коміків (8 коміків з рейтингом вище 6,5).

value = [1, 7] означає, що з цих 8 коміків, 1 отримає " NO ", а 7 отримають " GO ".

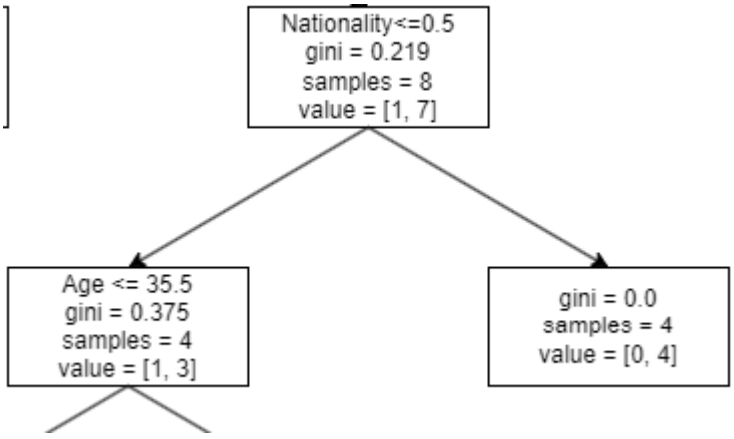



Рис. 2.2.4 Схематичний приклад роботи Decision tree. Частина 3.

rue - Продовжуються 4 коміки:

Age

Age <= 35,5 означає, що коміки віком 35,5 або молодші підуть по стрілці вліво, а решта підуть по стрілці вправо.

gini = 0,375 означає, що близько 37,5% вибірки підуть в одному напрямку.

samples = 4 означає, що в цій гілці залишилось 4 коміків (4 коміки з Великої Британії).

value = [1, 3] означає, що з цих 4 коміків 1 отримає " NO ", а 3 отримають " GO".

False - 4 коміки закінчуються тут:

gini = 0.0 означає, що всі зразки мають одинаковий результат.

samples = 4 означає, що в цій гілці залишилось 4 коміки (4 коміки не з Великої Британії).

value = [0, 4] означає, що з цих 4 коміків 0 отримає " NO ", а 4 отримають " GO".

True - 2 коміки закінчуються тут:

gini = 0.0 означає, що всі зразки мають одинаковий результат.

samples = 2 означає, що в цій гілці залишилося 2 коміки (2 коміки віком 35,5 або молодші).

value = [0, 2] означає, що з цих 2 коміків 0 отримає " NO ", а 2 отримають " GO".

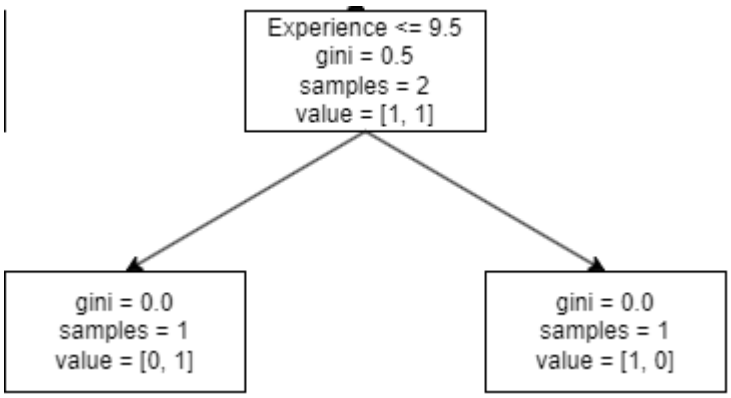
False - Продовжуються 2 коміки:

Досвід роботи

Досвід роботи <= 9,5 означає, що коміки з досвідом роботи 9,5 років або менше підуть по стрілці вліво, а решта підуть по стрілці вправо.

gini = 0,5 означає, що 50% вибірки підуть в один напрямок.

samples = 2 означає, що в цій гілці залишилося 2 коміки (2 коміки старше.




True - 1 Комік Закінчується Тут:

gini = 0.0 означає, що всі вибірки мають однаковий результат.

samples = 1 означає, що в цій гілці залишився 1 комік (1 комік з досвідом 9,5 років або менше).

value = [0, 1] означає, що 0 отримає "NO", а 1 отримає "GO".

False - 1 Комік Закінчується Тут:

gini = 0.0 означає, що всі вибірки мають однаковий результат.

samples = 1 означає, що в цій гілці залишився 1 комік (1 комік з досвідом більше 9,5 років).

value = [1, 0] означає, що 1 отримає "NO", а 0 отримає "GO".

Прогнозування значень

Ми можемо використовувати Дерево рішень для прогнозування нових значень.

Приклад: Чи повинен я піти на шоу з участю 40-річного американського коміка з 10-річним досвідом і рейтингом комедії 7?

1 означає "GO"

0 означає "NO"

Відповідь - 0, отже це означає "NO".

### *PostgrSQL*

В моделі прийняття рішення, окрім коду, є ще одна не менш важлива річь, це база даних, де данні накопичуються, звідки вони отримуються і де потім зберігаються після опрацювання моделлю. Такою базою є PostgrSQL.

PostgreSQL є безкоштовною та відкритою реляційною системою управління базами даних (RDBMS), яка наголошує на розширюваності та відповідності SQL. Вона була спочатку названа POSTGRES, що походить від її походження як наступник бази даних Ingres, розробленої в Університеті Каліфорнії, Берклі[29][30]. У 1996 році проект був перейменований на PostgreSQL, щоб відображати його підтримку SQL. Після перегляду в 2007 році, розробницька команда вирішила залишити назву PostgreSQL та псевдонім Postgres.

PostgreSQL має можливість виконання транзакцій з властивостями атомарності, послідовності, ізоляції та стійкості (ACID), автоматично оновлюваними видами, матеріалізованими видами, тригерами, зовнішніми ключами та збереженими процедурами[31]. Вона розроблена для роботи з різними завантаженнями, від одиночних машин до складів даних або веб-сервісів з багатьма паралельними користувачами. PostgreSQL була базою даних за замовчуванням для macOS Server та також доступна для Linux, FreeBSD, OpenBSD та Windows.

PostgreSQL керує конкурентністю за допомогою багатоверсійного управління конкурентністю (MVCC), що дає кожній транзакції "снапшот" бази даних, дозволяючи внесення змін без впливу на інші транзакції. Це в значній мірі усуває необхідність у читаючих блокуваннях та забезпечує дотримання принципів ACID. PostgreSQL пропонує три рівні ізоляції транзакцій: Читання комітів, Повторювані читання та Серіалізовані. Оскільки PostgreSQL є імунним до "dirty reads", запит на рівень ізоляції транзакції Read Uncommitted замість цього надає рівень read committed. PostgreSQL підтримує повну серіалізованість за допомогою методу серіалізованого знімку (SSI).

У PostgreSQL схема містить усі об'єкти, крім ролей та просторів таблиць. Схеми діють як простори імен, дозволяючи об'єктам з однаковим іменем існувати в одній базі даних. За замовчуванням відповідно до відновлення бази

даних є схема з іменем public, але можна додавати будь-які додаткові схеми, і схема public не є обов'язковою.

Параметр search_path визначає порядок, в якому PostgreSQL перевіряє схеми для об'єктів без вказаної схеми. За замовчуванням він встановлений на $user, public ($user відноситься до поточного користувача бази даних). Цей параметр можна встановити на рівні бази даних або ролі, але оскільки він є параметром сесії, його можна вільно змінювати (навіть декілька разів) під час сесії клієнта, що впливає лише на цю сесію.

Неіснуючі схеми, перераховані в search_path, при пошуку об'єктів, безпомилково пропускаються.

Нові об'єкти створюються в першій дійсній схемі (одній, яка існує в даний момент), яка з'являється в search_path..

Типи даних

Підтримуються широкі можливості нативних типів даних, включаючи:

Boolean

Arbitrary-precision numerics

Character (text, varchar, char)

Binary

Date/time (timestamp/time with/without time zone, date, interval)

Money

Enum

Bit strings

Text search type

Composite

HStore, an extension enabled key-value store within PostgreSQL

Arrays (variable-length and can be of any data type, including text and composite types) up to 1 GB in total storage size

Geometric primitives

IPv4 and IPv6 addresses

Classless Inter-Domain Routing (CIDR) blocks and MAC addresses

XML supporting XPath queries

Universally unique identifier (UUID)

JavaScript Object Notation (JSON), and a faster binary JSONB (not the same as BSON)

Крім того, користувачі можуть створювати власні типи даних, які, як правило, можуть бути повністю індексовані за допомогою інфраструктури індексування PostgreSQL - GiST, GIN, SP-GiST. Приклади таких типів даних включають географічну інформаційну систему (GIS) від проекту PostGIS для PostgreSQL.

Зовнішні обгортки даних

PostgreSQL може посилатися на інші системи для отримання даних через зовнішні обгортки даних (FDW). Це можуть бути будь-які джерела даних, такі як файлова система, інша система управління реляційними базами даних (RDBMS) або веб-сервіс. Це означає, що звичайні запити до бази даних можуть використовувати ці джерела даних як звичайні таблиці, навіть об'єднувати декілька джерел даних разом.

Інтерфейси

Для підключення до додатків PostgreSQL містить вбудовані інтерфейси libpq (офіційний інтерфейс додатків на C) та ECPG (вбудована система на мові C). Сторонні бібліотеки для підключення до PostgreSQL доступні для багатьох мов програмування, включаючи C++, Java, Julia, Python, Node.js, Go та Rust..

### *Power BI*

І остання частина моделі, це візуалізація, що полегшує роботу з даними, а також дає можливість людям, безпосередньо не задіяним в розробці, швидко отримати необхідну інформацію в доступній і зрозумілій їм формі. Я використовував BI систему під назвою Power BI.

Microsoft Power BI - це інтерактивний продукт для візуалізації даних, розроблений компанією Microsoft з основним фокусом на бізнес-інтелекті(BI)[32]. Він є частиною платформи Microsoft Power. Power BI - це колекція програмних сервісів, додатків та конекторів, які працюють разом, щоб перетворити не пов'язані джерела даних в зв'язні, візуально насичені та інтерактивні інсайти. Дані можуть бути введені шляхом прямого зчитування з баз даних, веб-сторінок або структурованих файлів, таких як електронні таблиці, CSV, XML та JSON. Power BI надає хмарні послуги бізнес-інтелекту, відомі як "Power BI Services", нарізі з інтерфейсом для робочого столу, який називається "Power BI Desktop". Він пропонує можливості складу даних, включаючи підготовку даних, виявлення даних та інтерактивні панелі управління[33]. У березні 2016 року Microsoft випустила додатковий сервіс, який називається Power BI Embedded на своїй хмарній платформі Azure. Одним з основних відмінностей продукту є можливість завантаження користувацьких візуалізацій.

Ключові компоненти екосистеми Power BI включають::

Power BI Desktop[34]

Microsoft Power BI Desktop - це програмне забезпечення для ПК та настільних комп'ютерів, призначене переважно для проектування та публікації звітів до служби Power BI. Power BI Desktop розроблено для аналітиків і поєднує сучасні інтерактивні візуалізації з провідними методами запиту та моделювання даних.

Power BI Service - це онлайн-сервіс на основі SaaS (програмне забезпечення як сервіс), раніше відомий як Power BI для Office 365, тепер - PowerBI.com або просто Power BI.

Power BI Mobile Apps - додатки для мобільних пристроїв Android і iOS, а також для телефонів та планшетів Windows.

Power BI Gateway - використовується для синхронізації зовнішніх даних в та з Power BI та необхідний для автоматичного оновлення. В режимі Enterprise також може використовуватися Power Automate (раніше - Flows) та PowerApps в Office 365.

Power BI Embedded - REST API Power BI може використовуватися для створення інформаційних панелей та звітів в звичайних додатках, які використовуються користувачами Power BI, а також не-Power BI користувачами.

Power BI Report Server - це розповсюджена на власному сервері Power BI звітність для компаній, які не можуть зберігати дані в хмарному сервісі Power BI Service або не бажають цього робити.

Power BI Premium - це пропозиція, що базується на потужності, яка дозволяє публікувати звіти широко по всій компанії без необхідності індивідуальної ліцензії для кожного користувача. Це забезпечує більшу масштабованість та продуктивність, ніж спільна потужність в Power BI Service.

Power BI Visuals Marketplace - це маркетплейс власних візуальних елементів та візуалізацій, що працюють на основі R. Це дає можливість оживити ваші бізнес-дані та отримати інсайти з результатів аналізу.

Power BI Dataflow - це Power Query-реалізація у хмарі, яку можна використовувати для трансформації даних з метою створення загального набору даних Power BI, який може бути доступний для кількох розробників звітів через Common Data Service від Microsoft. Це може бути альтернативою трансформації даних в SSAS, і забезпечити використання даних, що були трансформовані аналогічним чином, кількома розробниками звітів.

Power BI Dataset - це збірка даних, яка використовується в звітах Power BI та може бути підключена або імпортована до звіту Power BI. Dataset може бути підключений до одного або кількох Dataflow, щоб отримати вихідні дані.

Power BI Datamart

У Power BI, Datamart є контейнером, який поєднує Power BI Dataflows, набори даних та типу Data Mart або складу даних (у вигляді бази даних Azure SQL) в одному інтерфейсі. Інтерфейс може бути єдиним місцем для адміністрування як ETL-рівня (Dataflow), проміжного складу даних (зі

зберіганням зіркових схем, таблиць розмірностей, фактичних таблиць тощо), так і рівня моделювання (Dataset).

Power BI Datahub

Datahub - це центр даних для виявлення Power BI-наборів даних в службі Power BI організації, щоб набори даних можна було використовувати з одного центрального місця. Він надає інформацію про різні набори даних, а також точку доступу для роботи з ними, наприклад, для побудови звітів на їх основі, використання їх з функцією аналізу Excel, доступу до налаштувань, управління дозволами та іншого.

## 3. Структура інформаційної системи прийняття рішень та ключові процеси необхідні для її експлуатації

### 3.1. Логіка та алгоритм побудови програми системи прийняття рішень під час розгляду заявок на відновлення майна громадян України

Спочатку розберемо з яких елементів складеться система. Як вище було представлено це база даних PostgreSQL, модель написана на Python та PowerBI, як користувацький інтерфейс і аналітичний засіб.

То як працює система.


Рис. 3.1.1 Схема роботи системи

Як мі бачимо усі данні зберігаються в базі даних. В двох таблицях створених для зручності. Одна з навчальними даними, інша з випадково згенерованими даним на яких тестувалася система.

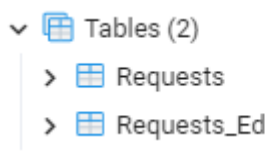



Рис. 3.1.2 Таблиці в базі даних.

Тобто в таблиці знаходяться заяви котрі мали б бути оброблені людиною і на їх основі модель може, в залежності від завдання, або передбачити, яка заявка буде прийнята, або по заданим параметрам відсіяти вірогідно помилкові заяви, чи такі, які мають менший пріоритет і їх важче задовольнити.

В іншій таблиці, не оброблені запити. Повинен зразу уточнити, декілька важливих моментів. Кожна заява з поміткою «rejected», не є видаленою і в обов’язковому порядку має бути перевірена людиною, за для опрацювання. Друге, створена мною архітектура бази даних є спрощеною за для тестування

моделі, і в реальних умовах не має бути такою же простою. А для коректної роботи моделі лише потрібна відмітка, про опрацювання, заяви, тобто ким вона була опрацьована, людиною чи моделлю.

Надалі розглянемо основну частину, тобто код.

Першим я продемонструю модуль який відповідає за з'єднання з базою даних.

**Config.py**

```
from configparser import ConfigParser


def config(filename='database.ini', section='postgresql'):
    # create a parser
    parser = ConfigParser()
    # read config file
    parser.read(filename)

    # get section, default to postgresql
    db = {}
    if parser.has_section(section):
        params = parser.items(section)
        for param in params:
            db[param[0]] = param[1]
    else:
        raise Exception('Section {0} not found in the {1} file'.format(section,
filename))

    return db
```

Конфігурація підключення.

**databas.ini**

```
[postgresql]
host=localhost
user=postgres
password=*********
database=Requests
```

Текстовий файл відповідає за збереження інформацію по підключенню. Тобто до якої бази підключатися, який пароль і інше.

Generator.py

```
 from msilib.schema import Class
import random
import names
import pandas
import psycopg2
from sqlalchemy import create_engine
import DTIMP
import numpy as np
from psycopg2.extensions import register_adapter, AsIs
psycopg2.extensions.register_adapter(np.int64, psycopg2._psycopg.AsIs)


alchemyEngine =
create_engine('postgresql+psycopg2://postgres:masterkey@localhost/Requests',
pool_recycle=3600)
dbConnection  = alchemyEngine.connect()
ed = pandas.read_sql("select \"RequestID\" from \"Requests\" ", dbConnection)
pandas.set_option('display.expand_frame_repr', False)

n = ed.shape[0]
df=ed['RequestID']
a=0
while a < n:
    id = df.iloc[a]
    a = a+1

def properrty_type_cr():
    number = str(random.randint(0, 5))
    if number == '2' or number == '1' or number == '0':
        property = 'Квартира'
    elif number == '3' or number == '4':
        property = 'Приватний будинок'
    else :
        property = 'Житлове приміщення'

    return property

def region():
    number = int(random.randint(0, 17))
    if  number == 1 or number == 0:
        region = 'Одеська область'
    elif number == 2 or number == 3:
        region = 'Дніпропетровська область'
    elif number == 4 or number == 5 or number == 6:
        region = 'Харківська область'
    elif number == 7 :
        region = 'Київська область'
```

```
    elif number == 8 :
        region = 'Запорізька область'
    elif number == 9 or number == 10:
        region = 'Миколаївська область'
    elif number == 11 or number == 12 or number == 13:
        region = 'Сумська область'
    elif number == 14 or number == 15 or number == 16:
        region = 'Чернігівська область'
    else :
        region = 'Львівська область'

    return region

def random_phone_num_generator():
    first = str(random.randint(100, 999))
    second = str(random.randint(1, 888)).zfill(3)

    last = (str(random.randint(1, 9998)).zfill(4))
    while last in ['1111', '2222', '3333', '4444', '5555', '6666', '7777', '8888']:
        last = (str(random.randint(1, 9998)).zfill(4))

    return '{}-{}-{}'.format(first, second, last)




for i in range(500):
    id = id + 1
    DTIMP.insert_request(id)
    property = properrty_type_cr()

    if property =='Квартира':
        Sqare = int(random.randint(10, 120))
        numres = int(random.randint(1, 12))

    elif property =='Приватний будинок':
         Sqare = int(random.randint(90, 400))
         numres = int(random.randint(1, 17))

    else:
         Sqare = int(random.randint(6, 700))
         numres = int(random.randint(1, 17))

    name= names.get_full_name()
    regions = region()
    phone = "+380-"+random_phone_num_generator()
    DTIMP.full_update_Request(id,name, property,Sqare, numres,('IS NULL'),regions,phone)
```

Даний модуль цілком відповідає за данні. Тобто він генерує запити імітуючи реальні заяви. Це було зроблено з декількох причин. Перша, я не маю доступу до реальних баз даних, а навіть маючі такі данні, я не мав би права розповсюджувати приватну інформацію. За для більш-менш реалістичності, я генерую імена, області, це за для невеликої аналітики, номера телефонів, і необхідні поля для роботи моделі.

За для розуміння нереалістичності даних, я використав англійську базу імен, а номера телефонів, автентичність збігається тільки з кодом країни +380.

```
import psycopg2
from Config import config

#  update Status used for inserting result into table

def update_Request(RequestID, Status):

    sql = """ UPDATE "Requests"
                SET  "Status" = %s
                WHERE "RequestID" = %s"""
    conn = None
    updated_rows = 0
    try:
        # read database configuration
        params = config()
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)
        # create a new cursor
        cur = conn.cursor()
        # execute the UPDATE  statement
        cur.execute(sql,  (Status, RequestID))
        # get the number of updated rows
        updated_rows = cur.rowcount
        # Commit the changes to the database
        conn.commit()
        # Close communication with the PostgreSQL database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
```

```
        if conn is not None:
            conn.close()

    return updated_rows

# modul created  for data generator

def full_update_Request(RequestID, Name, Property_Type, Sqare,  Number_of_Residents,
Address, Region_City, Contacts):

    sql = """ UPDATE "Requests"
                SET "Name" = %s,
                "Property_Type" = %s,
                "Sqare" = %s,
                "Number_of_Residents" = %s,
                "Address" = %s,
                "Region_City" = %s,
                "Contacts" = %s
                WHERE "RequestID" = %s"""
    conn = None
    updated_rows = 0
    try:
        # read database configuration
        params = config()
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)
        # create a new cursor
        cur = conn.cursor()
        # execute the UPDATE  statement
        cur.execute(sql, (Name,  Property_Type, Sqare, Number_of_Residents, Address,
Region_City, Contacts, RequestID))
        # get the number of updated rows
        updated_rows = cur.rowcount
        # Commit the changes to the database
        conn.commit()
        # Close communication with the PostgreSQL database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()

    return updated_rows
```

```
#this one is for generator too, it creates an id which will be updated by the previous
module

def insert_request(RequestID):
    """ insert a new vendor into the vendors table """
    sql = """INSERT INTO "Requests"("RequestID")
             VALUES(%s) RETURNING "RequestID";"""
    conn = None
    vendor_id = None
    try:
        # read database configuration
        params = config()
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)
        # create a new cursor
        cur = conn.cursor()
        # execute the INSERT statement
        cur.execute(sql, (RequestID,))
        # get the generated id back
        vendor_id = cur.fetchone()[0]
        # commit the changes to the database
        conn.commit()
        # close communication with the database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()

    return RequestID
```

Цей модуль відповідає за операції, пов’язані з безпосередньою роботою коду з базою за допомогою SQL запитів. Тобто він співпрацює з головнім модулем оновлюючи данні в колонці Status, за для внесення результатів роботи програми в базу. Та для внесення згенерованих даних в базу.

```
import pandas
from sklearn.tree import export_graphviz
import graphviz
import psycopg2
from sklearn import tree
```

```
from sklearn.tree import DecisionTreeClassifier
from sqlalchemy import create_engine
import DTIMP
import numpy as np
from psycopg2.extensions import register_adapter, AsIs
psycopg2.extensions.register_adapter(np.int64, psycopg2._psycopg.AsIs)


# connection to db and geting data. ed data used for lerning. df data which need to be
processed
alchemyEngine =
create_engine('postgresql+psycopg2://postgres:masterkey@localhost/Requests',
pool_recycle=3600)
dbConnection  = alchemyEngine.connect()
ed = pandas.read_sql("select \"Property_Type\", \"Sqare\", \"Number_of_Residents\",
\"Status\" from \"Requests_Ed\" ", dbConnection)
df = pandas.read_sql("select \"RequestID\", \"Property_Type\", \"Sqare\",
\"Number_of_Residents\", \"Status\" from \"Requests\" where \"Status\" is NULL",
dbConnection)
pandas.set_option('display.expand_frame_repr', False)

# giving a code number for used not numerical data in educational data
d = {'Квартира': 0, 'Приватний будинок': 1, 'Житлове приміщення': 2}
ed['Property_Type'] = ed['Property_Type'].map(d)
d = {'accepted': 1, 'rejected': 0}
ed['Status'] = ed['Status'].map(d)

# giving a code number for used not numerical data in un processed data
dd = {'Квартира': 0, 'Приватний будинок': 1, 'Житлове приміщення': 2}
df['Property_Type'] = df['Property_Type'].map(dd)
dd = {'accepted': 1, 'rejected': 0}
df['Status'] = df['Status'].map(dd)

features = ['Property_Type', 'Sqare', 'Number_of_Residents']

# make sure to use only correct data
idf = df['RequestID']
x = ed[features]
y = ed['Status']
s = df[features]


dtree = DecisionTreeClassifier()
dtree = dtree.fit(x, y)
n = s.shape[0]

i= 0
```

```
# module which update data in db and make prediction about designated rows
while i < n:
    row = pandas.DataFrame(s.iloc[[i]])
    id = idf.iloc[i]
    prediction = (dtree.predict(row))

    if prediction == 0:
        answer = 'rejected'
    else :
        answer = 'accepted'

    DTIMP.update_Request(id, answer)

    i = i+1

dot_data = export_graphviz(dtree, out_file=None,
                    feature_names=['Property_Type', 'Squre', 'Number_of_Residents'],
                    class_names=['rejected', 'accepted'],
                    filled=True, rounded=True,
                    special_characters=True)

graph = graphviz.Source(dot_data)
graph.render("decision_tree")

dbConnection.close()
```

І остання, головна часина, яка безпосередньо відповідає за збір необхідних даних, навчання моделі, а потім її експлуатації.

Перша частина коду відповідає за збір і підготовку відповідних датасетів, навчального та необробленого. Далі кожному параметру, потрібно надати числове значення щоб модель могла розпізнати параметри. Далі данні розділюються, на фітчі, та статус і цей процес повторюються як для навчальних даних, так для необроблених. А в кінці вже іде рішення, та трансформація рішення з цифрового виду в відповідне значення “rejected”, або “accepted”.

Технологія decision tree як і інші системи типу machine learning працює на основі навчання. Особливо це видно в використаній технології. Головним

аспектом для коректної роботи decision tree е данні використанні для навчання моделі і на їх основі вона буде робити свій вибір.

В реаліях даної моделі данні для навчання можуть бути штучно створені записи, та данні реальних заявок опрацьованих людиною. Чим краще створенні навчальні данні, тим точнішою буде модель. Змінивши навчальні данні можна змінити її роботу.

Я штучно створив набір даних, який покриває більшу частину ситуацій, як позитивних так і негативних, коли данні точно помилкові, або не мають пріоритетності. Цей дата сет виключно суб'єктивній, але через те що система доволі гнучка, можна просто змінити навчальній сет, на той який потрібен, а також якщо необхідно як зменшити кількість параметрів, так і змінити ціль системb, на аналітично-передбачувальний.

Табл. 3.1.1 Приклад навчальних даних

| Property_Type | Sqare | Number_of_Residents | Status | Confirmd By |
|---|---|---|---|---|
| Приватний будинок | 100 | 7 | accepted | H |
| Приватний будинок | 100 | 1 | rejected | H |
| Приватний будинок | 100 | 3 | accepted | H |
| Приватний будинок | 100 | 5 | accepted | H |
| Квартира | 50 | 7 | rejected | H |
| Квартира | 50 | 5 | accepted | H |
| Квартира | 50 | 3 | accepted | H |
| Квартира | 50 | 1 | rejected | H |

| | | | | |
|---|---|---|---|---|
| Житлове приміщення | 30 | 3 | rejected | H |
| Житлове приміщення | 30 | 4 | rejected | H |
| Житлове приміщення | 30 | 5 | rejected | H |
| Житлове приміщення | 30 | 6 | rejected | H |

Це невеликий витяг з існуючих даних для навчання. Як можна побачити, типи власності існують всього трьох типів. Першим, треба оговорити Житлове приміщення. Як я казав раніше, під категорію житлове приміщення, може підходити великий спектр приміщень, які відповідають нормам житлового кодексу та санітарнім нормам. З цієї причини я вирішив що уся нерухомість з поміткою житлове приміщення має бути опрацьовано виключно людиною, або мають бути додані додаткові параметри, до яких я не маю доступу чи о яких не маю інформації. Наприклад бази пошкоджень, для звіряння, фотографічних матеріалів, та іншого.

В інших типах, головним фактором, є відношення, кількості людей, до типу нерухомості і до площі. Наприклад якщо в квартирі 50 метрів квадратних і в заяві указано що там проживає 7 людей, то це порушення мінімальних рекомендацій що до житлової площі, і або помилкове, або має бути розглянуто але окремо, не в першу чергу. І таким же чином прописано для кожного типу нерухомості, з середньою площею.

Ще одним головним аспектом в навчанні цієї моделі, є проблема перенавчання. Якщо в навчальному датасеті, наприклад усі прийняті заявим будуть, квартирами, то може статися так що модель буде вважати параметр “квартира”, головним фактором того, що заява була прийнята.

| RequestID [PK] numeric (15) | Name character varying (100) | Property_Type character varying (20) | Sqare numeric (5) | Number_of_Residents numeric (4) | Address character varying (200) | Status character varying (10) | Region_City character varying (40) | Contacts character varying (20) |
|---|---|---|---|---|---|---|---|---|
| 2 | Kristi Alley | Приватний будинок | 160 | 11 | IS NULL | rejected | Запорізька область | +380-230-322-3540 |
| 3 | Bobby Burkhead | Житлове приміщення | 419 | 0 | IS NULL | rejected | Харківська область | +380-943-064-4154 |
| 4 | Norene Germany | Квартира | 26 | 10 | IS NULL | rejected | Сумська область | +380-415-808-1836 |
| 5 | James Carter | Квартира | 105 | 11 | IS NULL | rejected | Львівська область | +380-666-572-1835 |
| 6 | Keith Hunt | Приватний будинок | 275 | 2 | IS NULL | rejected | Харківська область | +380-941-525-9601 |
| 7 | Eileen Stamper | Приватний будинок | 242 | 6 | IS NULL | accepted | Миколаївська область | +380-529-577-8400 |
| 8 | Daniel Hedgpeth | Житлове приміщення | 393 | 0 | IS NULL | rejected | Чернігівська область | +380-337-489-7916 |
| 9 | Jonathan Do | Приватний будинок | 217 | 9 | IS NULL | accepted | Миколаївська область | +380-600-065-1724 |
| 10 | Numbers Ballard | Приватний будинок | 117 | 4 | IS NULL | accepted | Сумська область | +380-242-752-4654 |
| 11 | Charles Knapp | Приватний будинок | 194 | 5 | IS NULL | accepted | Миколаївська область | +380-927-255-6284 |
| 12 | Ron Conklin | Житлове приміщення | 464 | 0 | IS NULL | rejected | Одеська область | +380-416-638-0778 |
| 13 | Claudine Culp | Квартира | 48 | 5 | IS NULL | accepted | Сумська область | +380-850-273-7511 |
| 14 | Linda Stollsteimer | Житлове приміщення | 52 | 0 | IS NULL | rejected | Харківська область | +380-602-851-2422 |
| 15 | Eric Williams | Квартира | 20 | 7 | IS NULL | rejected | Запорізька область | +380-462-536-5749 |
| 16 | Carol Mccullough | Приватний будинок | 234 | 1 | IS NULL | rejected | Харківська область | +380-994-794-7967 |
| 17 | Jazmin Hillin | Приватний будинок | 254 | 11 | IS NULL | accepted | Миколаївська область | +380-814-591-3908 |
| 18 | Leonard Arimoto | Квартира | 50 | 8 | IS NULL | rejected | Львівська область | +380-751-325-4930 |
| 19 | Nicholas Rodriguez | Житлове приміщення | 352 | 0 | IS NULL | rejected | Миколаївська область | +380-617-017-5912 |
| 20 | Roman Oliver | Житлове приміщення | 483 | 0 | IS NULL | rejected | Одеська область | +380-866-189-8742 |
| 21 | Joyce Bailey | Приватний будинок | 280 | 10 | IS NULL | accepted | Київська область | +380-196-614-7609 |
| 22 | Frances Coulter | Приватний будинок | 237 | 10 | IS NULL | accepted | Харківська область | +380-557-865-1688 |
| 23 | Albert Adamski | Квартира | 112 | 8 | IS NULL | rejected | Одеська область | +380-506-407-1727 |
| 24 | Dave Kun | Квартира | 36 | 4 | IS NULL | accepted | Чернігівська область | +380-248-566-6663 |
| 25 | Kathrine Pope | Приватний будинок | 268 | 1 | IS NULL | rejected | Миколаївська область | +380-816-425-1526 |
| 26 | Tara Merritt | Приватний будинок | 318 | 17 | IS NULL | accepted | Сумська область | +380-771-063-3079 |
| 27 | Shelley Reese | Квартира | 101 | 2 | IS NULL | rejected | Миколаївська область | +380-348-154-4619 |
| 28 | Jamie Gower | Житлове приміщення | 147 | 0 | IS NULL | rejected | Чернігівська область | +380-320-556-2794 |
| 29 | Dianne Sullivan | Квартира | 64 | 3 | IS NULL | accepted | Харківська область | +380-618-355-8276 |

Рис. 3.1.3 Оброблені запити

У додаток потрібно продемонструвати схему за якою працює decision tree. Тобто дерево рішення.

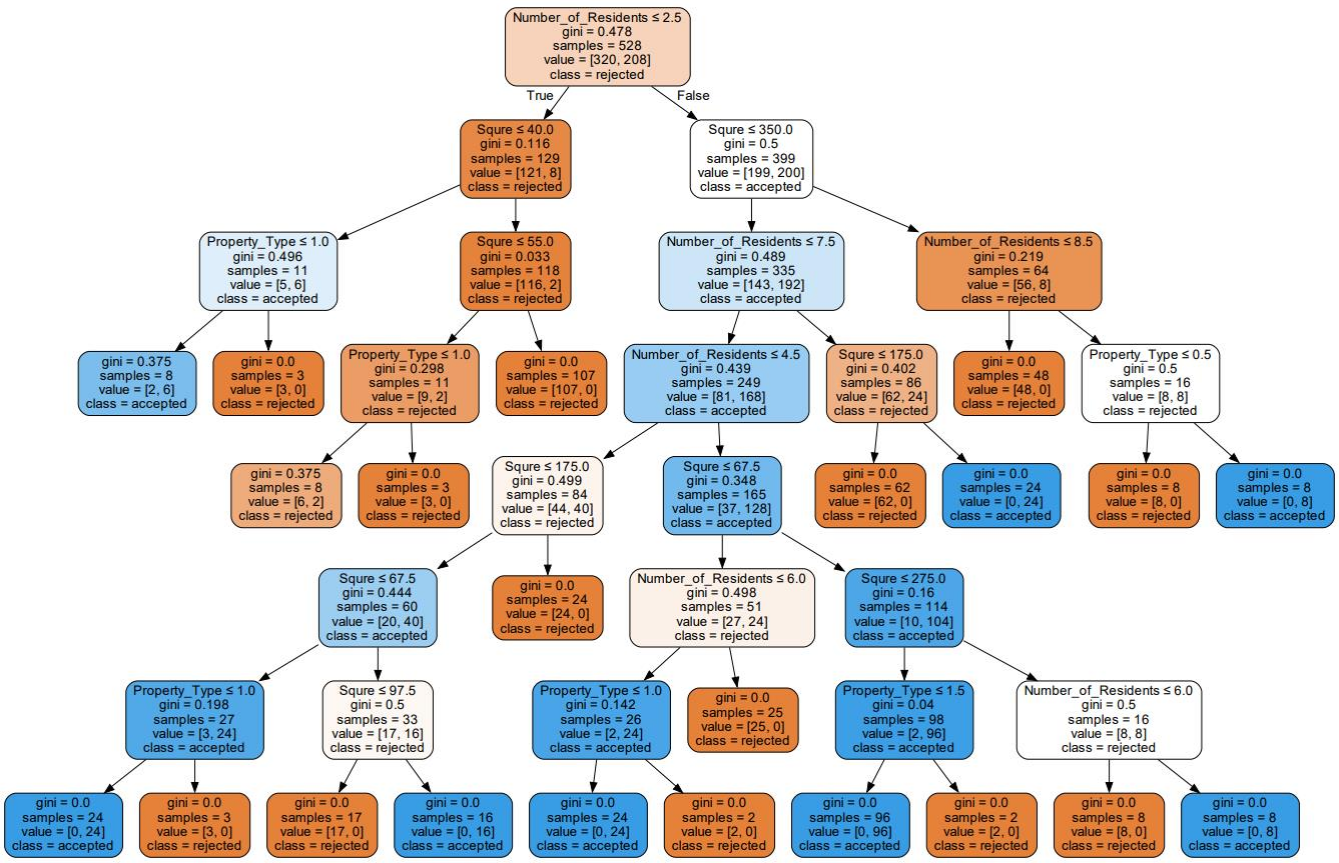



Рис. 3.1.4 Дерево рішення

Останньою частиною моделі, є її інтерфейс, про який було написано вище. Подібні BI системи є необхідними для користувачів, за для кращого розуміння даних. І їх використання людям, які не мають відношення до розробки даної моделі. А також швидкого і зручного доступу до необхідної інформації, з можливістю наочного зображення, а також зручної аналітики.

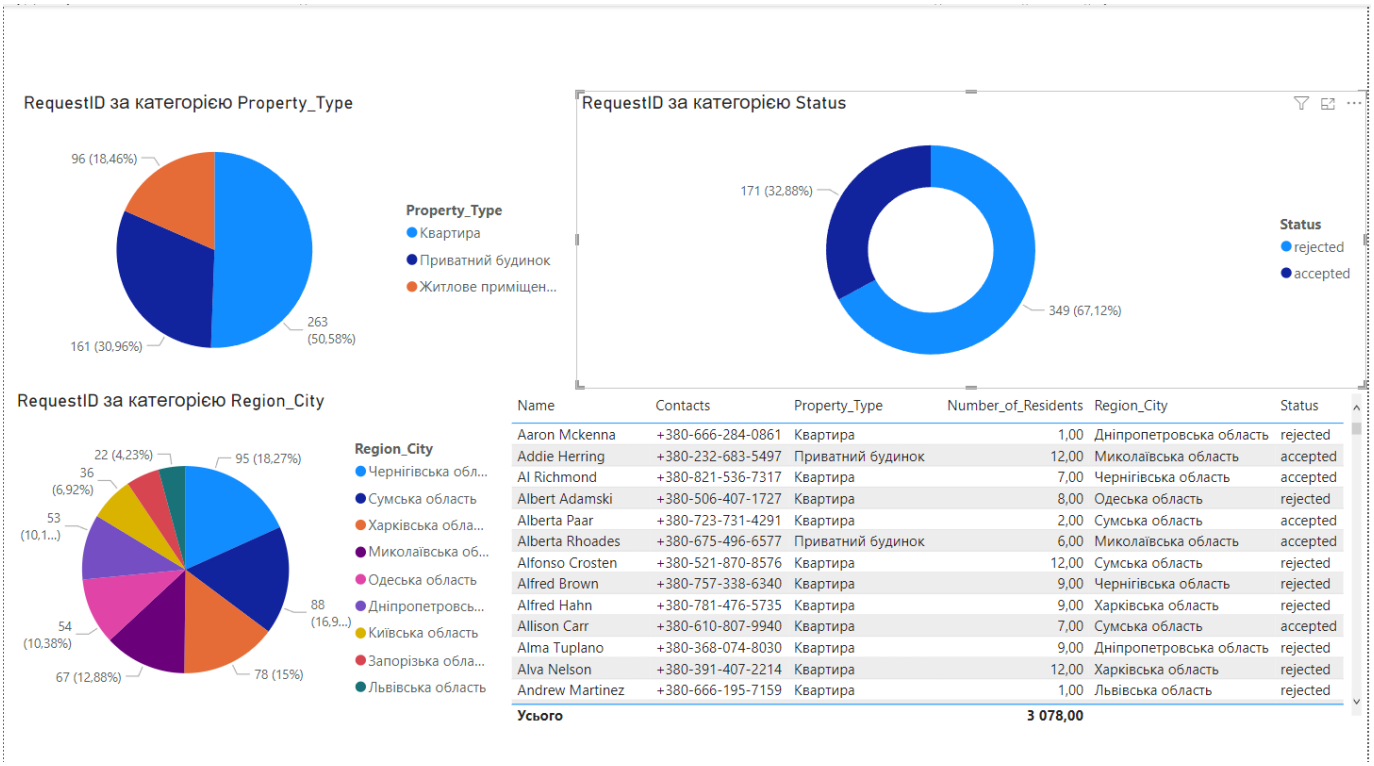



Рис. 3.1.5 Дашборд в PowerBI

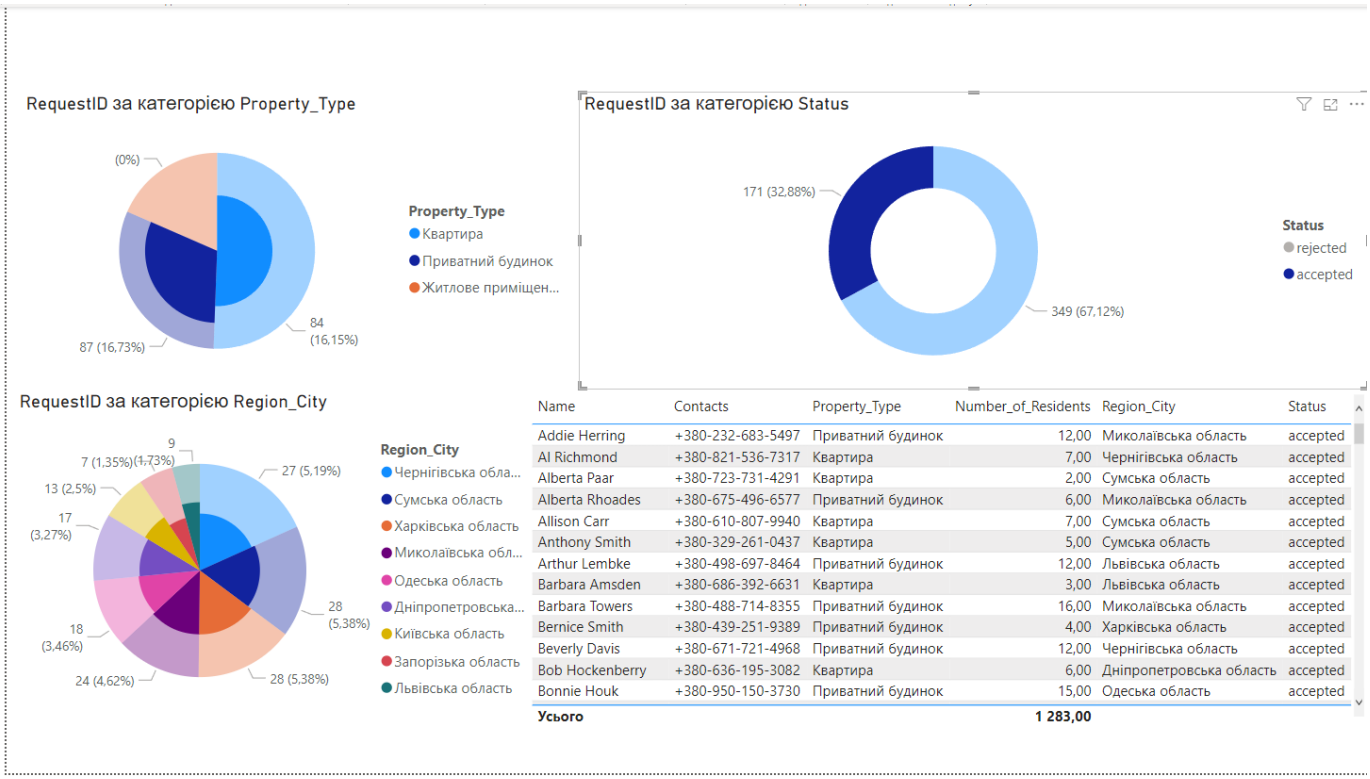



Рис. 3.1.6 Аналітика PowerBI

Це приклад, коли вибрані всі прийняті заяви. Як можна побачити з дашборду. Ми можемо провести детальну аналітику, по кожному необхідному показнику. З прийнятих заяв, які були згенеровані, найбільша кількістю заяв прийнято з Харківської області та Сумської області. Прийнятих заяв з типу нерухомості, квартири та приватні будинки набрали майже однакову кількість прийнятих заяв. І зразу видно кількість постраждалих.

PowerBI отримує данні постійно направляючи запити в реальному часі за допомогою технології, тому всі зміні будуть там відображені.

### 3.2. Ключові процеси необхідні для коректної експлуатації системи ухвалення рішень

Ключові процеси, необхідні для коректної експлуатації системи прийняття рішень, включають:

Збір та зберігання даних: Система прийняття рішень повинна мати можливість збирати та зберігати дані, які будуть використовуватися для прийняття рішень. Ці дані можуть включати інформацію про минулі транзакції, клієнтів, продукти, конкурентів, ринки тощо.

Обробка та аналіз даних: Дані, зібрані системою, повинні бути оброблені та проаналізовані, щоб виявити тенденції, закономірності та зв'язки. Важливо, щоб система прийняття рішень могла використовувати різні методи та алгоритми аналізу даних, такі як статистичний аналіз, машинне навчання та штучний інтелект.

Візуалізація даних: Система прийняття рішень повинна мати можливість представляти дані в зрозумілому та наочному вигляді, щоб користувачі могли швидко та легко розуміти результати аналізу.

Прийняття рішень: На основі аналізу даних та представленої інформації користувачі системи прийняття рішень повинні приймати рішення. Важливо,

щоб система прийняття рішень мала можливість автоматично приймати рішення відповідно до заздалегідь визначених правил та критеріїв.

Моніторинг результатів: Після прийняття рішення система прийняття рішень повинна моніторити результати та дії, які були здійснені. Це допомагає оцінити ефективність рішення та коригувати процес прийняття рішень у майбутньому.

Оновлення системи: Система прийняття рішень повинна постійно оновлюватися та удосконалюватися, щоб враховувати зміни в бізнес-процесах, зміни на ринку та нові технології та методи аналізу даних.

## Висновки

На даний момент, це лише модель, яка може виконувати свої функції, але не в повному обсязі в якому необхідно для вирішення такої комплексної проблеми. Щоб вона пішла в роботу, необхідна консультація з спеціалістами, та більш складна система, щоб витримати навантаження великої кількості заяв. А також інтеграція в систему де вона буде використовуватися.

Зараз, модель є виключно тестовою і надалі потребую доопрацювання, виходячи зі специфіки її використання та поставлених завдань.

На даний момент, її можливо зробити дещо складнішою, створивши градацію (класифікацію), відносно пошкоджень, пріоритетності, але більш розвернутої.

Також з даних які попередньо опрацювала система, можна використати для проектування майбутньої моделі встановлення. Таким чином приблизно спрогнозувати, необхідну кількість будівель, їх розміри, затрати та інше.

Для кращої точності, модель потребує набагато більше навчальних даних та більш різноманітних, які би покривали різні ситуації.

Тобто програма потребую подальшого розроблення, але вже на місць її подальшого використання.

## Список використаних джерел


1. Keen, Peter (1980). "Decision support systems : a research perspective". Cambridge, Massachusetts : Center for Information Systems Research, Alfred P. Sloan School of Management.
2. Sprague, R;(1980). "A Framework for the Development of Decision Support Systems." MIS Quarterly. Vol. 4, No. 4, pp.1-25.
3. Power, D. J. (2002). “Decision support systems: concepts and resources for managers.” Westport, Conn., Quorum Books.
4. Haettenschwiler, P. (1999). “Neues anwenderfreundliches Konzept der Entscheidungsunterstützung. Gutes Entscheiden in Wirtschaft, Politik und Gesellschaft.” Zurich, vdf Hochschulverlag AG: 189-208.
5. Sprague, R. H. and E. D. Carlson (1982). “Building effective decision support systems.” Englewood C⌙liffs, N.J., Prentice-Hall. ISBN 0-13-086215-0
6. Power, D. J. (1996). “What is a DSS?” The On-Line Executive Journal for Data-Intensive Decision Support 1(3).
7. J.Donavan, S.Madnick(1977). “Institutional and ad hoc DSS and their effective use” p. 1-10
8. Holsapple, C.W., and A. B. Whinston. (1996). “Decision Support Systems: A Knowledge-Based Approach.” St. Paul: West Publishing. ISBN 0-324-03578-0
9. Richard D Hackathorn, Peter GW Keen (1981/9/1) Journal MIS quarterly “Organizational strategies for personal computing in decision support systems” p. 21-27
10. Wright, A; Sittig, D (2008). "A framework and model for evaluating clinical decision support architectures q". Journal of Biomedical Informatics. P**.**982–990
11. Zhang, S.X.; Babovic, V. (2011). "An evolutionary real options framework for the design and management of projects and systems with complex real options and exercising conditions". Decision Support Systems. P**.**119–129.

12. Salvaneschi, Paolo; Cadei, Mauro; Lazzari, Marco (1996). "Applying AI to structural safety monitoring and evaluation". IEEE Expert. p.24–34. Retrieved 5 March 2014.
13. Masera, Alberto; et al. "Integrated approach to dam safety". Comitê Brasileiro de Barragens. Retrieved 16 December 2020.
14. Lancini, Stefano; Lazzari, Marco; Masera, Alberto; Salvaneschi, Paolo (1997). "Diagnosing Ancient Monuments with Expert Software" p. 288–291
15. Постанова Кабінету Міністрів України від 26 березня 2022 р. № 380 «Про збір, обробку та облік інформації про пошкоджене та знищене нерухоме майно внаслідок бойових дій, терористичних актів, диверсій, спричинених військовою агресією Російської Федерації»
16. Жданенко, Ірина (27 вересня 2019). Дія – Держава і Я – бренд цифрової держави!
17. Президент, Прем'єр-міністр, Мінцифри презентували мобільний застосунок "Дія". www.kmu.gov.ua. Департамент комунікацій Секретаріату Кабінету Міністрів України. 6 лютого 2020. Процитовано 28 березня 2021.
18. Diia Summit: нові цифрові послуги та документи. thedigital.gov.ua. 5 жовтня 2020.
19. Diia Summit 2.0: які революційні послуги тепер доступні українцям у Дії. diia.gov.ua. 17 травня 2021.
20. Rossum, Guido Van (20 January 2009). "The History of Python: A Brief Timeline of Python"
21. Peterson, Benjamin (20 April 2020). "Python Insider: Python 2.7.18, the last release of Python 2".
22. The Cain Gang Ltd. "Python Metaclasses: Who? Why? When?"
23. "PyDatalog". Archived from the original on 13 June 2020. Retrieved 22 July 2012.
24. Peters, Tim (19 August 2004). "PEP 20 – The Zen of Python". Python Enhancement Proposals. Python Software Foundation.
25. Venners, Bill (13 January 2003). "The Making of Python". Artima Developer. Artima.
26. Martelli, Alex; Ravenscroft, Anna; Ascher, David (2005). Python Cookbook, 2nd Edition. O'Reilly Media. p. 230.

27. von Winterfeldt, Detlof; Edwards, Ward (1986). "Decision trees". Decision Analysis and Behavioral Research. Cambridge University Press. pp. 63–89.
28. Kamiński, B.; Jakubczyk, M.; Szufel, P. (2017). "A framework for sensitivity analysis of decision trees". Central European Journal of Operations Research. 135–159.
29. Stonebraker, M.; Rowe, L. A. (May 1986). The design of POSTGRES. Proc. 1986 ACM SIGMOD Conference on Management of Data. Washington, DC. Retrieved December 17, 2011.
30. "PostgreSQL: History". PostgreSQL Global Development Group.
31. "What is PostgreSQL?". PostgreSQL 9.3.0 Documentation. PostgreSQL Global Development Group.
32. "Bring your data to life with Microsoft Power BI". Microsoft.com. Microsoft.
33. "Magic Quadrant for Business Intelligence and Analytics Platforms". Gartner.com. Gartner, Inc.
34. Hyman, Jack (8 February 2022). Microsoft Power BI for dummies. NJ: wiley. pp. 47–62.