

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій**

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«29» 05 _____ 2023 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інформаційних систем» спеціальності 121 «Інженерія програмного
забезпечення»
на тему: «Система виявлення об'єктів будівництва з проблемним станом»

Виконав (-ла):

слухач (-ка) IV курсу, групи ЗП-01

Кононов Максим Анатолійович _____

Керівник:

Кандидат технічних наук, доцент

Лісовиченко О. І. _____

Рецензент:

доц. каф. ІСТ, к.т.н., доц.

Остапченко Костянтин Борисович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів без
відповідних посилань.

Слухач (-ка) _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«15» 05 _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу слухачу
Кононову Максиму Анатолійовичу

1. Тема роботи «Система виявлення об'єктів будівництва з проблемним станом», керівник роботи Лісовиченко Олег Іванович, кандидат технічних наук, доцент затверджені наказом по університету від «22» 05 _____ 2023 р. № 1873
2. Термін подання слухачем роботи 13.06.2023 _____
3. Вихідні дані до роботи
4. Зміст пояснювальної записки
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

6. Дата видачі завдання 01.03.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Узгодження теми дипломної роботи	01.03.2023-05.03.2023	
2	Збір інформації про предметну галузь	10.03-2023-17.03.2023	
3	Вибір інструментів для реалізації проекту	18.03-2023-20.03.2023	
4	Формування навчальних і тестових вибірок	22.03-2023-25.03.2023	
5	Розробка веб-сторінки для роботи з даними	26.03-2023-15.04.2023	
6	Реалізація серверної частини проекту	17.04.2023-01.05.2023	
7	Тестування моделей	04.05.2023-10.05.2023	
8	Формування висновків	11.05.2023-20.05.2023	
9	Написання тексту дипломної роботи	22.05.2023-13.06.2023	

Слухач

Максим КОНОНОВ

Керівник роботи

Олег ЛІСОВИЧЕНКО

АНОТАЦІЯ

Кононов М. А. Система виявлення об'єктів будівництва з проблемним станом. КПІ ім. Ігоря Сікорського, Київ, 2023.

Робота містить 70 с. тексту, 5 рисунків, посилання на 25 літературні джерела, додатки.

Ключові слова: аналіз даних, будівництво, веб-додаток, випадковий ліс, логістична регресія, машинне навчання, метод опорних векторів, програмування.

Об'єктом розробки є система виявлення проблемних об'єктів будівництва.

Мета розробки – підвищення ефективності існуючих рішень для аналізу даних про житлову нерухомість.

У дипломній роботі розроблено веб-додаток, який працює з інформацією про об'єкти будівництва, розподіляє її на тестові та навчальні вибірки для подальшого аналізу засобами власного скрипту, а також сторонніх програм. Приділено велику увагу рішенням проблем виявлення неуспішних конструкцій житлової сфери на певному етапі їх створення. Виконано ретельний аналіз ефективності новітніх інструментів мови програмування Python для роботи з інформацією, розв'язано задачі систематизації та візуалізації даних.

Отримані результати можуть бути корисними при автоматизації систем для аналізу даних в будівельній галузі.

SUMMARY

Kononov M. A. A system for detecting construction objects in a problematic state. Igor Sikorsky KPI, Kyiv, 2023.

The project contains 70 pages text, 5 figures, links to 25 literary sources, annexes.

Keywords: construction, data analysis, logistic regression, machine learning, programming, random forest, support vector machine, web application.

The object of development is the system for detecting construction objects in a problematic state.

The purpose of the development is the increasing of effectiveness of existing solutions for real estate data analysis.

The graduation project developed a web application that works with information about construction objects, divides it into test and training samples for further analysis by means of our own script, as well as third-party programs. Much attention is paid to solving the problems of identifying unsuccessful constructions in the residential sphere at a certain stage of their creation. A thorough analysis of the effectiveness of the latest tools of the Python programming language for working with information was performed, and the problems of data systematization and visualization were solved.

The obtained results can be useful in the automation of data analysis systems in the construction industry.

**Пояснювальна записка
до дипломної роботи
на тему: «Система виявлення об'єктів будівництва з
проблемним станом»**

Київ – 2023 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	4
ВСТУП	5
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ	8
1.1 Поняття будівництва.....	8
1.2 Єдина державна електронна система у сфері будівництва.....	9
1.3 Юридичні положення щодо об'єктів будівництва з проблемним станом ...	12
1.4 Запобігання виникненню об'єктів незавершеного будівництва	13
Висновок до розділу.....	17
2 СИСТЕМА ПРОГНОЗУВАННЯ ОБ'ЄКТІВ НЕЗАВЕРШЕНОГО БУДІВНИЦТВА	19
2.1 Опис використаних технологій.....	19
2.2 Огляд доступних джерел даних.....	23
2.3 Опис веб-проекту для роботи з об'єктами будівництва	25
2.4 Опис використаних моделей.....	31
Висновок до розділу.....	34
3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	36
3.1 Розгортання та тестування веб-проекту для роботи з нерухомістю	36
3.2 Зовнішній скрипт для аналізу даних про об'єкти будівництва.....	39
3.3 Оптимальний клас складності, верифікація окремих моделей	43
3.4 Графічне представлення навчальних даних	45
3.5 Оцінка якості реалізованих рішень	49
Висновок до розділу.....	50

					ЗПІ-зп01.080БАК.005 ПЗ							
		№ докум.	Підпис									
Розробив					Система виявлення об'єктів будівництва з проблемним станом Пояснювальна записка				Літ.	Арк.	Аркушів	
Перевірів		Лісовиченко О.І.							Г		2	70
Затв.									КПІ ім. Ігоря Сікорського Група ЗПІ-зп01			

4 ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ ВІДНОВЛЕННЯ ОБ’ЄКТІВ

БУДІВНИЦТВА З ПРОБЛЕМНИМ СТАНОМ	51
4.1 Базові поняття.....	51
4.2 Опис реальної системи масового обслуговування	53
4.3 Тестування розробленої мережі.....	59
4.4 Система прогнозування результатів відновлення об’єктів будівництва з проблемним станом.....	61
Висновок до розділу.....	64
ВИСНОВКИ.....	66
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТОК А.....	71

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ЄДЕСБ – Єдина державна електронна система у сфері будівництва

ООП – об'єктно-орієнтоване програмування

ОС – операційна система

ПЗ – програмне забезпечення

СКБД – система керування базами даних

СМО – система масового обслуговування

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Будівництво є однією з найважливіших галузей людської діяльності. Воно вважається одним з видів матеріального виробництва, в якому створюються готові до експлуатації будівлі, споруди та їх комплекси.

У реальному житті інколи трапляються ситуації, коли на певному етапі проектування, зведення, здачі в експлуатацію або розподілення житлової площі об'єкт будівництва стає незавершеним або незаселеним. Це призводить до величезних збитків компаній, які виконують роботи, сприяють їх банкрутству. Висока ймовірність виникнення недобудов призводить до того, що потенційні покупці житла значно рідше вкладають гроші в нерухомість на стадії створення.

Виникає замкнуте коло: забудовник намагається знизити власні витрати та зменшити ймовірність заморожування будівництва за рахунок отримання якомога більше коштів від потенційних клієнтів до завершення робіт, проводячи інтенсивні рекламні компанії, спокушаючи покупців житла нижчими цінами, але поступово підвищуючи їх по мірі збільшення ступеня готовності споруди для експлуатації. Клієнт, який згоден переплатити, в свою чергу, покладається на забудовника, рідше приймає фінансові ризики на себе, сподіваючись, що будівництво, профінансоване належним чином сторонніми організаціями, буде успішно завершене. Але такі сподівання рідко стають реальністю, що дуже часто призводить до недостатнього фінансування на певному етапі створення нерухомості. В результаті цього ми отримуємо некрасиву споруду на одній з мальовничих вулиць населеного пункту, яку можна спостерігати навіть на далекій відстані. Повне відновлення таких будинків трапляється рідко.

Названа проблема має місце в багатьох містах та селищах нашої держави. Наприклад, за статистикою 2020 року, тільки в Києві налічується 70 недобудов, що є значною кількістю для міста-мільйонника. У зв'язку із тенденцією спадання економічного розвитку ця кількість постійно зростає. Наявність та широка розповсюдженість даної проблеми пояснює **актуальність** нашої роботи, зумовлює необхідність глибоких досліджень на цьому поприщі..

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		5

До проблем будівництва відноситься й відсутність даних про незавершені об'єкти в більшості міст України. Тому систематизація будь-якої інформації про них, а також створення різних інформаційних проектів у даній галузі може сприяти суттєвому покращенню ситуації в майбутньому. Це може виявитись корисним у сферах житлово-комунального господарства, сприяти розвитку будівельної галузі в будь-якій державі. Поява достовірної інформації про недобудови в певному конкретному населеному пункті може покращити репутацію органів місцевої влади або представників приватного бізнесу, врятувати екологію шляхом зменшення надмірних витрат матеріальних ресурсів на будівництво, зберегти кошти забудовників, їх партнерів та кінцевих споживачів нерухомості, сприяти руйнуванню схем незаконного збагачення в даній галузі. Проблема невдалих будівельних споруд є дуже широкою і полягає не тільки в їх розповсюдженості, а й в дефіциті достовірної детальної інформації, після отримання якої можна залучити фінансові, матеріальні та людські ресурси для відновлення. Все це доводить актуальність і необхідність даної роботи.

Ми можемо стверджувати, що **об'єктами** нашого дослідження є недобудови, оскільки саме їх наявність, а також відсутність легкодоступних довідкових даних про них породжує проблему, яку ми маємо вирішити шляхом реалізації цього проекту.

Предметом дослідження є система виявлення об'єктів будівництва з проблемним станом. Це поняття є багатогранним і може включати в себе як способи збору, класифікації та впорядкування наявних даних, так і проблеми розробки програмних продуктів для вирішення задач передбачення проблемності певної конструкції на певних етапах її створення.

Метою роботи є створення ПЗ для виявлення об'єктів, будівництво яких в майбутньому може бути призупинене або зовсім відмінене. Додатково ми ставимо перед собою ціль належним чином випробувати існуючі інструменти для аналізу даних, створити рішення для моделювання процесів відновлення конструкцій на основі певних параметрів. Такий додаток повинен мати лише рекомендаційний характер та надавати наближені дані користувачу.

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

Для цього ми маємо вирішити наступні **завдання**:

1. проаналізувати ситуацію на ринку нерухомості в різних містах нашої держави, отримати якомога більше інформації як про завершені, так і про неуспішні об'єкти будівництва;
2. на основі наявних даних сформулювати критерії, аналіз яких допоможе встановити, чи стане новий об'єкт будівництва проблемним;
3. вивчити інструменти аналізу даних для прогнозування значень на основі заданої інформації, обрати найбільш придатні для нашого дослідження;
4. ознайомитись із технологіями, на основі яких базуватиметься програма для виявлення об'єктів будівництва з проблемним станом;
5. розробити веб-додаток, який відображає та обробляє завантажені дані, здійснює прогнози стану нових будівель;
6. поглибити знання в області імітаційного моделювання, дослідити роботу системи масового обслуговування;
7. на основі дослідженої СМО створити другий програмний продукт для імітації процесу відновлення невдалих будівельних конструкцій;
8. провести численні експерименти з різними даними та параметрами, надати детальну інформацію про їх результати, зробити висновки.

Одержані результати досліджень можуть мати велике **практичне значення** в будівельній, житлово-комунальній, господарчій, законодавчій сферах, сприяти вдосконаленню існуючих інструментів для роботи з даними, надати потенційним покупцям нерухомості раніше недоступні довідкові статистичні дані.

Апробація розроблених рішень здійснюється шляхом автоматизованої побудови статистичних даних у вигляді таблиць, графіків, діаграм разом із відповідними висновками. До цього процесу також відноситься аналіз ефективності використаних інструментів, створення приміток щодо результатів досліджень, можливі рекомендації майбутнім власникам житла.

Дипломна робота складається з наступних розділів: вступ, основні розділи, висновки, список основних джерел із 25 найменувань, 1 додаток. Загальний обсяг складає 70 сторінок.

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Предметна область даної дипломної роботи включає в себе загальні відомості про будівельну галузь, існуючі інформаційні ресурси з детальними достовірними даними про об'єкти нерухомості. Власниками таких сервісів є як приватні компанії, так і державні органи влади. Також ми маємо висвітлити поняття недобудови, розглянути законодавчі аспекти роботи з нею, приділити певну увагу причинам виникнення об'єктів будівництва з проблемним станом, розглянути способи протидії їх виникнення на юридичному, матеріально-технічному та фінансовому рівнях, описати деякі популярні шахрайські схеми.

1.1 Поняття будівництва

Будівництво – галузь матеріального виробництва, в якій створюються основні фонди виробничого і невиробничого призначення: готові до експлуатації будівлі, споруди, їх комплекси [1].

В термінології профільного Міністерства розвитку громад та територій «будівництво» – це нове будівництво, реконструкція, технічне переоснащення, реставрація та капітальний ремонт об'єктів будівництва [3].

Термін «будівництво» охоплює:

– будівельні роботи, серед них земляні роботи і спорудження, конструктивні зміни, реставраційні роботи, капітальний і поточний ремонт (куди входять чистка й пофарбування) та знесення усіх видів будинків чи будівель;

– цивільне будівництво, куди входять земляні роботи й спорудження, конструктивні зміни, капітальний і поточний ремонт та знесення, наприклад, аеропортів, доків, гаваней, внутрішніх водних шляхів, гребель, захисних споруд на берегах річок і морів поблизу зон обвалів, автомобільних доріг і шосе, залізниць, мостів, тунелів, віадуків та об'єктів, пов'язаних з наданням послуг, комунікації, дренаж, каналізація, водопостачання й енергопостачання;

– монтаж та демонтаж конструкцій з елементів заводського виробництва, а також виробництво збірних елементів на будівельному майданчику.

– Розрізняють такі галузі будівництва: промислове, транспортне, житлово-цивільне. До видів відносяться великопанельне, збірно-монолітне каркасне, цегляне, дерев'яне житлове, панельно-каркасне.

Вирізняються види будівель та споруд, які відносяться до одного з класів наслідків (відповідальності) здійснює проектна організація за погодженням із замовником будівництва [5].

До незначних наслідків (СС1) належать такі об'єкти: індивідуальні (садибні) житлові будинки, садові, дачні будинки не вище двох поверхів (без урахування мансардного поверху) з площею до 500 квадратних метрів, господарські будівлі та споруди, гаражі, а також інші будівлі й споруди, які не належать до класів наслідків СС2 та СС3.

До середніх наслідків (СС2) належать об'єкти: характеристики можливих наслідків від відмови яких перевищують певний рівень можливої небезпеки для здоров'я і життя певної кількості людей, які постійно або періодично перебувають на об'єкті.

До значних наслідків (СС3) належать такі об'єкти як • пам'ятки культурної спадщини, об'єкти підвищеної небезпеки, будівлі з висотою понад 100 метрів або з рівнем можливої небезпеки для здоров'я і життя понад 400 осіб.

Після завершення будівництва забудовник має подати декларацію про готовність до експлуатації об'єкта з незначними наслідками або отримати сертифікат, який підтверджує прийняття в експлуатацію об'єктів класу наслідків СС2 та СС3 [14].

1.2 Єдина державна електронна система у сфері будівництва

У нашій державі функціонує реєстр будівельної діяльності, який є комплексною базою даних, яка поєднує інформацію про об'єкти будівництва, учасників будівельного процесу, відомості про дозвільні документи, містобудівну та проектну документацію, експертизи проектів, відомості про технічну інвентаризацію та багато іншої інформації, що раніше зберігалася в розрізних реєстрах та базах даних [17].

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

З 2022 року в Україні діє Єдина державна електронна система у сфері будівництва (ЄДЕСБ) – це загальнонаціональна інформаційна система, що покликана впорядкувати процес будівництва в Україні і через максимальну публічність інформації зробити його прозорим та вільним від корупції. Передбачається, що на порталі будуть доступні всі послуги, що пов'язані з будівництвом, включаючи отримання дозволів. Старий реєстр ДАБІ було інтегровано до Системи як архівну складову.

Портал Єдиної система в сфері будівництва є публічною частиною самої Системи та має на меті забезпечення безкоштовного безперешкодного доступу до інформації, що створюється в системі для всіх типів користувачів. На даному етапі Портал забезпечує зручний пошук інформації з можливістю проектування даних на карту, а також базові аналітичні інструменти. Також на порталі доступні корисні мережеві сервіси, які стануть в нагоді як професіоналам в цій сфері так і новачкам. Функціональність Порталу буде розширюватися по мірі розвитку Єдиної системи в сфері будівництва [15].

Єдина державна електронна система у сфері будівництва (скорочено - ЄДЕСБ)- Загальнонаціональна інформаційна система, що покликана впорядкувати процес будівництва в Україні і через максимальну публічність інформації зробити його прозорим та вільним від корупції. ЄДЕСБ має реалізувати весь життєвий цикл будівництва об'єкту починаючи з отримання містобудівних умов та обмежень, закінчуючи його введенням в експлуатацію. Передбачається, що більшість інформації та документації необхідної для ведення будівництва буде створюватися в Системі відповідними суб'єктами, що дозволить централізувати та стандартизувати дані з багатьох розрізнених джерел інформації. Власником Системи виступає держава в особі Міністерства розвитку громад, територій та інфраструктури України. Технічним адміністратором виступає ДП «Дія».

В розумінні Системи об'єкт будівництва є основою її функціонування. Це будівля, споруда або її частина, що знаходиться у процесі створення, реконструкції, реставрації або капітального ремонту або ж прийнята в

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

експлуатацію. Для кожного об'єкта будівництва присвоюється унікальний номер (ідентифікатор), за яким завжди легко знайти всю історію по об'єкту. Вам більше не потрібно пам'ятати номери різних документів, коли і ким вони були видані, адже всі вони будуть прив'язані до ідентифікатору об'єкту будівництва.

Системи складається з трьох компонентів:

1. реєстр будівельної діяльності – ядро Системи, яке містить інформацію про об'єкти будівництва, учасників будівництва, дозвільні та інші документи;
2. портал Системи – публічний Портал, на якому розміщено інформацію, що міститься у реєстрі будівельної діяльності, аналітика роботи системи тощо;
3. електронний кабінет Системи – внутрішня частина, яка надає змогу чиновникам, організаціям та атестованим особам вносити та реєструвати документи, підтверджувати участь у будівництві та відслідковувати його статус.

Для органів місцевого самоврядування доступ надається уповноваженим органам містобудування та архітектури до наступних наборів даних:

1. дозвільні документи;
2. містобудівні умови і обмеження;
3. будівельні паспорти.

Система знаходиться в захищеному середовищі в дата центрі Деново з підтвердженою відповідністю щодо захисту інформації. Перед запуском системи було проведено відповідні навантажувальні та тести на вразливості. Всі дії користувачів в системі фіксуються, а документи підписуються електронним підписом. Для більш надійного зберігання даних та громадського контролю за їх незмінністю в Системі передбачається запровадження елементів block chain.

Відомості, які містяться в електронній системі, є відкритими і загальнодоступними, окрім реєстраційних номерів облікових карток платників податків, паспортних даних, місця проживання фізичної особи, договорів про надання послуг, документів, поданих фізичними та юридичними особами для отримання послуг.

Наразі до Системи підключені:

– органи державного архітектурно-будівельного контролю;

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

- місцеві органи містобудування та архітектури;
- саморегулювальні організації у сфері архітектурної діяльності та атестовані ними особи;
- проектні та експертні організації;
- органи з присвоєння адрес;
- органи охорони культурної спадщини;
- правоохоронні органи;
- виробники будівельної продукції;
- аудитори у сфері енергетики;
- нотаріуси;

1.3 Юридичні положення щодо об'єктів будівництва з проблемним станом

Право власності на новостворене нерухоме майно (житлові будинки, будівлі, споруди тощо) виникає з моменту завершення будівництва (створення майна).

Якщо договором або законом передбачено прийняття нерухомого майна до експлуатації, право власності виникає з моменту його прийняття до експлуатації.

Якщо право власності на нерухоме майно відповідно до закону підлягає державній реєстрації, право власності виникає з моменту державної реєстрації.

До завершення будівництва особа вважається власником матеріалів, обладнання тощо, які були використані в процесі цього будівництва (Стаття 331 Цивільного кодексу України) [14].

Порядок придбання об'єкта незавершеного будівництва за договором купівлі-продажу, міни зводиться до [18]:

1. укладання договорів власниками матеріалів та обладнання;
2. реєстрації права власності відповідним органом.

Для державної реєстрації права власності на об'єкт незавершеного будівництва подаються [13]:

- документ, що посвідчує речове право на земельну ділянку;
- документ, що надає право на виконання будівельних робіт;
- документ, що містить опис об'єкта незавершеного будівництва за результатами технічної інвентаризації.

Також слід отримати всю документацію по землі та об'єкту будівництва: дозвільну документацію, проект договору на підписання, документи на підтвердження повноважень особи, яка буде підписувати договір.

Покупцями об'єктів незавершеного будівництва можуть бути:

- громадяни України, іноземні громадяни;
- юридичні особи, зареєстровані на території України;

Приватизація об'єктів незавершеного будівництва здійснюється відповідними органами, у тому числі за участю уповноважених ними юридичних осіб. У такому разі право власності переходить до покупця після сплати ціни в повному обсязі.

1.4 Протидія виникненню об'єктів незавершеного будівництва

Придбання квартири на стадії будівництва досі залишається дуже ризиковою справою, тому що більшість будівельних афер відбуваються на початковій стадії цього процесу. При виборі об'єкта інвестиції слід розглядати будинки готовністю понад 50%. Але навіть надійна та успішна будівельна компанія не може повністю гарантувати здачу будинку в обіцяні терміни.

Надійність та успішність забудовника відіграє важливу роль при виборі об'єкта інвестування, оскільки ці характеристики залежать від вдалості реалізації проекту будівництва. Щоб переконатися в цьому, інвестор може самостійно перевірити надану інформацію про забудовника, надану відділом продажу. У цьому разі інвестор може використовувати цілий арсенал доступних джерел цінної інформації, до яких відносяться [2]:

- єдиний державний реєстр юридичних та фізичних осіб-підприємців – за його допомогою можна дізнатись склад учасників, забудовника, статутний фонд

підприємства, керівника, дату заснування підприємства-забудовника, можливість ліквідації найближчим часом (перебування у стадії банкрутства);

– державний реєстр обтяжень – з нього інвестор може дізнатись про будь-які арешти, заборони відчуження майна (зокрема і об'єктів будівництва);

– державний реєстр іпотек – цей документ дозволяє переконатись, чи не знаходиться в заставі (іпотеці) об'єкт будівництва.

Не зайвим буде перевірити й існуючі (або неіснуючі) судові справи, учасником яких виступає забудовник. Такі відомості можна отримати з Єдиного державного реєстру судових рішень, доступ до якого є відкритим. Для цього достатньо зайти на сайт www.reyestr.court.gov.ua та заповнити відповідні поля.

Отримана таким чином інформація дасть змогу виявити факти, які прямо чи опосередковано можуть впливати на можливість забудовника закінчити будівництво та здати будинок в експлуатацію. Наявність судового позову банку із забудовником про стягнення значної заборгованості, оскарження права власності чи права користування земельною ділянкою для будівництва, внутрішні корпоративні суперечки, зазвичай, негативно впливають на репутацію. Зрозуміло, що за наявності обтяжень інвестору необхідно задуматись над тим, чи варто вкладати гроші в таку компанію.

Корисною для інвестора також буде інформація про будівельні (інвестиційні) проекти, які вже були реалізовані забудовником. У цьому випадку не варто нехтувати неофіційною інформацією з Інтернету. Тут можна використовувати новини, тематичні форуми, журнали про новобудови. Навіть незважаючи на те, що отриману таким шляхом інформацію не завжди можна перевірити на достовірність, на тематичних ресурсах інвестор може знайти для себе чимало цікавого. Наприклад, дізнатися про те, чи є конфлікти забудовника з іншими інвесторами чи мешканцями навколишніх будинків тощо [24].

Важливим питанням, яке необхідно підняти та проаналізувати інвестору, є правовий режим земельної ділянки, на якій проводиться або планується будівництво. З цією метою необхідно запитати в забудовника необхідні правовстановлюючі документи. Особливу увагу слід звертати на договори оренди

землі. Оскільки оренда – це володіння та користування землею протягом обмеженого терміну, слід перевірити, коли він закінчується. Він має бути не меншим за термін, заявлений забудовником у документах для добудови об'єкта. Якщо термін будівництва більший за термін оренди, то існує серозний, обґрунтований ризик, що забудовнику не вдасться продовжити оренду на новий строк. У зв'язку з цим будівництво може бути заморожене, в чому не зацікавлений інвестор.

Крім документів на землю, слід також перевірити дозвоільні документи на будівництво. Ними є декларація чи дозволи, зареєстровані у місцевій архітектурно-будівельній інспекції. За відсутності у забудовника одного із зазначених документів, він не має законних підстав розпочинати будівельні роботи, а створений без таких документів об'єкт визнається самовільним будівництвом разом і тягне за собою певні негативні наслідки.

Схеми придбання житла можуть бути різні, тому договори, які укладаються з інвесторами, також матимуть свої юридичні особливості. Деякі загальні положення, які є в майже усіх договорах купівлі-продажу квартир [25]:

- розрахунки за житло: інвестор повинен чітко розуміти, яку суму він має заплатити забудовнику чи механізм (формула) її отримання;

- порядок оформлення права власності на квартиру: кінцевою метою придбання квартири в будинку, який ще не завершений, є оформлення на інвестора (покупця) правовстановлюючих документів, на підставі яких бюро технічної інвентаризації буде зобов'язане зареєструвати право власності, тому у договорі має бути чітко прописані обов'язки забудовника (і терміни їх виконання) щодо передачі інвестору необхідних документів для реєстрації права власності;

- відповідальність забудовника за затримку здачі житлового будинку в експлуатацію.

Також не слід забувати про проведення ретельної ревізії тексту договору на предмет наявності прихованих платежів.

Серйозна будівельна компанія завжди зацікавлена у швидкому проведенні робіт з метою дотримання графіку і вчасної здачі об'єкта в експлуатацію. Тому

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

бажано відвідати будівельний майданчик для переконання в тому, що будівельні роботи на ньому ведуться досить активно. Не завадить також і поговорити з самими будівельниками, розпитати їх про графік та умови виконання робіт, наявність претензій до керівництва і т.п. Якщо працівникам невчасно виплачують заробітну плату, то існує великий ризик затримки будівництва [25].

Одним із способів уникнення шахрайських схем є придбання житла на вторинному фонді, оскільки найбільшу кількість ризиків містить в собі купівля від забудовника. Дуже часто трапляються ситуації, коли компанії зупиняють будівництво, заморожують процес. Таким чином покупці стають ошуканими співниками. Якщо цей страх великий, краще звернутись до вторинного ринку. Сьогодні цей сегмент починає стабілізуватись, тому слід придивитись саме до готового житла [24].

Вирішити проблему недобудов в Україні допоможе кластерний підхід. Кластер – це група географічно сусідніх взаємопов'язаних компаній (постачальники, виробники та ін.) та пов'язаних з ними організацій (освітні організації, органи державного управління, інфраструктурні компанії), які функціонують у певній сфері та взаємодоповнюють одна одну. За даними експертів, зараз кластери є однією з ефективних форм організації регіонального інноваційного розвитку, за якою на ринку конкурують уже не окремі підприємства, а цілі комплекси, які скорочують свої витрати завдяки спільній технологічній кооперації компаній [2].

Для цього галузь потребує законодавчої бази, яка б гарантувала захист прав вкладників. Для того, щоб у нас у майбутньому не було недобудов (зараз їх налічується понад 3 тисяч) та обдурених вкладників, ми повинні більш активно співпрацювати з регіональними відділами так, щоб муніципалітет також брав участь, а відповідальність несли власники землі та банки. Для захисту власника потрібно створити відповідні закони, які б гарантували вкладникам, що їхні гроші, вкладені в будівництво, не пропадуть [25].

Висновок до розділу

У даній частині дипломної роботи ми проаналізували предметну область, головними поняттями якої є «будівництво», «об'єкт з проблемним станом», законодавча взаємодія з ним, інформаційно-довідкові ресурси в нашій країні, способи протидії виникненню невдалих споруд.

Термін «будівництво» охоплює все, що пов'язано із створенням житла. Важливим аспектом є класифікація нерухомих об'єктів за 3 класами наслідків:

Нами була описана єдина державна електронна система у сфері будівництва (ЄДЕСБ), яка створена для надання прозорості інформації про об'єкти будівництва, а також сприяння протидії корупції. Робота сервісу регулюється спеціально прийнятими для цього законами. Цей сервіс складається з реєстру будівельної діяльності, публічного порталу, на якому розміщена вся інформація про існуючі будівельні споруди та електронного кабінету, через який відповідні організації здійснюють реєстрацію об'єктів. Така система є частиною проекту «Дія», запущена у вересні 2020 року.

Нами був описаний термін «об'єкт незавершеного будівництва», новітні законодавчі аспекти взаємодії з ним (порядки реєстрації, придбання, приватизації, необхідні документи для цього), а також способи порятунку ошуканих інвесторів шляхом залучення державних органів влади.

Також були описані заходи для запобігання виникненню об'єктів будівництва з проблемним станом. До них відносяться:

1. перевірка забудовника на законодавчому рівні шляхом залучення державних реєстрів, які містять достовірні дані про юридичних та фізичних осіб-підприємців, їх обтяжень, іпотек, судових справ, документів на земельну ділянку, дозволи на виконання робіт тощо;
2. оформлення договірної документації належним чином;
3. аналіз даних про вже реалізовані проекти;
4. відвідування будівельного майданчика для встановлення рівня доброчесності виконавця по відношенню до його працівників та обіцяних термінів завершення робіт;

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		17

5. придбання житла на вторинному ринку після завершення всіх будівельних процесів та здачі в експлуатацію;

6. розробка законодавчих механізмів для залучення муніципалітетів та надання фінансових гарантій вкладникам;

Основними документами для проведення більшості законних операцій із нерухомими об'єктами проблемного стану є:

– декларації на початок будівельних робіт та про наявність або відсутність порушень технічних характеристик споруди (висота, кількість секцій і т. п.);

– ліцензія компанії-забудовника, технічні умови на тимчасове підключення до електромереж;

– документи на земельну ділянку;

– відомості про компанію забудовника в Єдиному державному реєстрі юридичних осіб та фізичних осіб-підприємців.

Відповідно до чинного законодавства покупцями недобудов можуть бути громадяни та юридичні особи, зареєстровані як на території України, так і в інших державах.

На теперішній час в Україні відомо про наявність понад 3000 об'єктів будівництва з проблемним станом, 70 з них знаходяться в Києві.

Проаналізувавши юридичний та інформаційно-технічний «фон» даної проблеми ми можемо стверджувати, що, починаючи з 2020 року державні органи влади виконують певні дії для покращення ситуації в будівельній сфері шляхом публікації даних про учасників цього виду діяльності та про їх об'єкти, що сприяє боротьбі з корупцією. Приймаються відповідні закони для більш гнучкої взаємодії забудовників та інвесторів, вирішення все більшої кількості їх проблем в юридичній площині. Проте питання повернення коштів у разі потрапляння в аферу ще досі залишається відкритим. Також слід відмітити незручність або відсутність можливості отримання даних про конкретні недобудови.

2 СИСТЕМА ПРОГНОЗУВАННЯ ОБ'ЄКТІВ НЕЗАВЕРШЕНОГО БУДІВНИЦТВА

Головна ціль нашої дипломної роботи – розробка системи виявлення або прогнозування об'єктів нерухомості з проблемним станом.

Ми маємо отримати програму для ручного та автоматичного введення даних про існуючі та майбутні об'єкти будівництва, збереження відомостей про них у базі даних, аналіз, редагування, доповнення, візуалізація та прогнозування стану майбутніх будинків шляхом використання декількох моделей. Тобто додаток має працювати за принципами CRUD (create, read, update and delete). Функція визначення стану має викликатись через веб-інтерфейс – результати мають потрапляти у відповідні комірки таблиці. Проведене дослідження повинне дати відповідь лише на одне запитання: чи буде майбутній об'єкт будівництва успішним, тобто чи буде він введений в експлуатацію без затримок та чи не виявиться збитковим для інвесторів в майбутньому.

Для реалізації проекту використовуються сучасні веб-технології, зокрема мови HTML, CSS, JavaScript та фреймворк Python під назвою Django, а також система керування базами даних PostgreSQL версії 15.3.2.

2.1 Опис використаних технологій

HTML (англ. HyperText Markup Language – мова розмітки гіпертексту) – стандартизована мова розмітки документів для перегляду вебсторінок у браузері. Браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора [23].

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації зі Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

Документ HTML 5.2 складається з трьох частин:

- декларація типу документа (англ. Document type declaration, Doctype), на початку документа, в якій визначається тип документа (DTD);
- шапка документа (знаходиться в межах елемента head), в якій записано загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;
- тіло документа (може знаходитися в елементі body), в якому міститься основна інформація документа.

Для перегляду HTML-коду документа використовується будь-який текстовий редактор, а для відображення результатів розмітки – браузер.

CSS (англ. Cascading Style Sheets, укр. Каскадні таблиці стилів) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних [22].

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстання та інші аспекти вигляду сторінки. Одна з головних переваг – можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових

браузерах та ін.). Отже, один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS.

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування. Найчастіше вона використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. За її допомогою вирішуються такі основні завдання [4, 20]:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових та прогресивних веб-застосунків (React, AngularJS, Vue.js);
- програмування на боці сервера (Node.js);
- мобільних застосунків (React Native, Cordova)
- сценаріїв в прикладних програмах (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter)
- всередині PDF-документів тощо.

Для роботи з базою даних ми використовуємо об'єктно-реляційна система керування базами даних (СКБД) **PostgreSQL** останньої версії 15.3-2. Вона є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite). Сервер PostgreSQL написаний на мові С. Зазвичай розповсюджується у вигляді набору текстових файлів із початковим кодом.

Django (Джанго) – високорівневий відкритий Python-фреймворк (програмний каркас) для розробки веб-систем. Названо його було на честь джазмена Джанго Рейнхардта (відповідно до музичних смаків одного зі засновників проєкту) [12]. Проте для нашого проєкту найбільшу цінність в Django представляє собою можливість використовувати бібліотеки Python для аналізу даних, зокрема NumPy, pandas, sklearn, matplotlib.

NumPy – це розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих

математичних функцій для операцій з цими масивами. NumPy можна розглядати як гарну вільну альтернативу MATLAB, оскільки мова програмування MATLAB зовні нагадує NumPy: обидві вони інтерпретовані, і обидві дозволяють користувачам писати швидкі програми поки більшість операцій проводяться над масивами або матрицями, а не над скалярами [8].

Pandas – програмна бібліотека, написана для мови програмування Python для маніпулювання даними та їхнього аналізу. Основні можливості [9]:

- об'єкт DataFrame із вбудованим індексуванням для маніпулювання даними;
- інструменти для зчитування та записування даних між структурами даних у пам'яті та різними форматами файлів;
- вирівнювання даних та вбудована підтримка пропущених даних;
- зміна формату для отримання зведених наборів даних;
- отримання зрізів за мітками, індексування з розширеними можливостями та отримання піднаборів з великих наборів даних [6];
- вставляння та вилучення стовпчиків у структурах даних;
- рушій групування, що дозволяє робити з наборами даних операції розділення-зміни-об'єднання (англ. split-apply-combine);
- злиття та з'єднання наборів даних;
- ієрархічне індексування осей для роботи з даними високої вимірності в структурі даних нижчої вимірності;
- функціональність для часових рядів: породження діапазонів дат та перетворення частоти, статистики рухливого вікна, лінійні регресії рухливого вікна, зсування дат та запізнювання.

Цю бібліотеку сильно оптимізовано за продуктивністю, критичні ланцюжки коду написано мовами Cython та C [9].

Scikit-learn (також відомий як sklearn) пропонує різноманітний набір статистичних моделей та машинного навчання. На відміну від більшості модулів, sklearn розробляється на Python, а не на C. Незважаючи на те, що він розроблений

на Python, ефективність sklearn пояснюється використанням NumPy для високопродуктивної лінійної алгебри та операцій з масивами [10].

Matplotlib – це бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується). Отримувані зображення можуть бути використані як ілюстрації в публікаціях. Користувач може вказати осі координат, сітку, додати підписи і пояснення, використовувати логарифмічну шкалу або полярні координати. Нескладні тривимірні графіки можна будувати з допомогою набору інструментів mplot3d. Існують й інші набори інструментів: для картографії, для роботи з Excel, утиліти для GTK. За допомогою Matplotlib можна створювати й анімовані зображення [7].

Бібліотека **Seaborn** – це пакет Python, який дозволяє нам створювати інфографіку на основі статистичних даних. Оскільки він виготовлений поверх Matplotlib, він є сумісним з ним. Крім того, він підтримує структуру даних NumPy та Pandas, так що нанесення графіків можна здійснювати безпосередньо з цих колекцій. З цим інструментом слід працювати в таких основних випадках:

- використовуються статистичні дані часових рядів, які потрібно побудувати із зображенням невизначеності навколо оцінок;
- необхідно візуально встановити різницю між двома підмножинами даних;
- потрібно зобразити одновимірні та двовимірні розподіли;
- необхідно візуалізувати моделі машинного навчання за допомогою лінійної регресії з незалежними та залежними змінними.

Візуалізація складних даних – це одна з найважливіших речей, про які піклується дана бібліотека. Seaborn може спростити речі, які важко досягти за допомогою Matplotlib [11].

2.2 Огляд доступних джерел даних

Для побудови системи машинного навчання, яка прогнозуватиме проблемність певного об'єкта будівництва за наявною про нього інформацією, ми маємо здобути історичні дані про вдалі та невдалі будинки. Чим більше існуючих об'єктів ми проаналізуємо, тим точніше працюватиме наша програма для певного

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

регіону, селища або всієї країни. На основі зібраних даних ми маємо сформувати та оцінити ознаки, які впливають на те, чи перетвориться проект на недобудову. Доступність інформації про проблемні об'єкти будівництва залежить від того, наскільки сильну увагу держава приділяє сфері інформаційних технологій, чи висвітлює вона дані про будівельні об'єкти в офіційних джерелах або чи проявляють таку ініціативу приватні компанії. Ми спробуємо надати та проаналізувати дані про житло у нашій країні, зокрема в місті Києві. Столиця – це одне з не багатьох міст, про об'єкти будівництва якої можна знайти багато інформації. Зібраних даних може бути достатньо для функціонування певної моделі машинного навчання з метою передбачення стану майбутнього житла.

На офіційному сайті Київської міської ради є актуальний перелік проблемних об'єктів будівництва (об'єктів самочинного будівництва та довгобудів) [6]. Він зроблений у вигляді таблиці, яка по кожному району надає дані про адресу будівлі, її замовника, відомість про наявність або відсутність вихідних даних на проектування та дозволу на виконання будівельних робіт, інформацію про підключення до інженерних мереж та про наявні причини зупинення будівельних процесів, у тому числі на підставі судових рішень або приписів ДАБІ, правовстановлюючі документи на земельну ділянку, поточний стан об'єктів тощо. Більш детальну інформацію про кожну будівлю, яка знаходиться у місті Києві та деяких прилеглих до нього селах, можна знайти на порталі Київської нерухомості [16]. Даний ресурс надає детальні дані, які включають в себе номер проекту, рік завершення або початку будівництва, матеріали, кількість поверхів, висоту стелі, фото.

Потужним недоліком існуючих веб-сайтів з інформацією про об'єкти будівництва є відсутність можливості отримати дані в зручному для аналізу вигляді, тобто як файл таблиці або бази даних. Тому для побудови системи машинного навчання або просто збереження та аналізу існуючих даних потрібно виконувати ручний увід дуже великої кількості записів. У зв'язку з цим нами було прийнято рішення ввести вручну лише інформацію про майже всі проблемні об'єкти житлового будівництва, зазначені на вищевказаному офіційному порталі

Київської міської ради, а дані про завершені будівлі згенерувати випадковим чином. Введені дані «накладаються» на реальні адреси, перелік яких можна завантажити у зручному табличному вигляді з офіційних сайтів деяких поштових сервісів. Для автоматизації цього процесу нами був розроблений спеціальний скрипт на мові Python, який завантажує всю таблицю адрес із CSV-файлу та відомості про невдалі об'єкти в базу даних PostgreSQL.

2.3 Опис веб-проекту для роботи з об'єктами будівництва

Для взаємодії з базою даних будівельних конструкцій нами був створений веб-проект засобами фреймворку Django. Він розташований у папці WebProject, яка міститься на GitHub або іншому носії інформації. Його розгортанню та детальному тестуванню присвячений наступний розділ даної дипломної роботи.

Проект має дуже просту структуру, після виконання певних маніпуляцій усі необхідні файли з кодом знаходяться в директорії PythonEnvironment\realestate. Якщо в цій же папці перейти в realestate, можна побачити файл **settings.py**. Він містить інформацію про веб-додаток та параметри бази даних, які включають в себе ім'я користувача, пароль, назву рушія, мережеві дані про адресу та порт і т. п. Наступним важливим файлом є urls.py, який містить дані про шляхи (назви запитів), які користувач «породжує» у процесі взаємодії з програмою. До них відносяться наступні пункти:

1. " – початкова адреса, на яку користувач потрапляє відразу при завантаженні сторінки. При цьому здійснюється повернення результатів SQL-запиту "SELECT * from \"Області\"" (ми отримуємо перелік областей з БД);

2. 'GetOneColumn/' – отримання інформації про вміст одного рядка шляхом виконання запиту до БД, сформованого frontend-засобами;

3. 'GetSeveralColumns/' - отримання вмісту декількох рядків шляхом виконання запиту до БД, створеного інструкціями JavaScript;

4. 'UpdateDatabase/' – оновлення бази даних інформацією, отриманою від frontend-частини проекту;

5. 'DeleteFromDatabase/' – видалення одного або декількох записів з БД;

6. 'predict/' – здійснення прогнозу стану майбутніх об'єктів будівництва.

Програмний код для виконання вищезазначених запитів до рушія розміщується у файлі **views.py**.

Веб-додатки Django отримують доступ і керують даними через об'єкти Python, які називаються моделями. Вони визначають структуру збереженої інформації, включаючи типи полів, їх максимальні розміри, значення за замовчуванням, параметри списку вибору, текст довідки для документації. Все це міститься у файлі **models.py**. У процесі розробки ми не побачили необхідності в його широкому використанні, оскільки параметри полів містяться в БД PostgreSQL. тому для спрощення проекту ми обмежились лише кількома рядками, які описують тільки одне поле *districts* ("Назва області").

Веб-сторінка представлена файлом **index.html**. Вона містить код розмітки на мові HTML 5, стильового оформлення (CSS 3), програмні реалізації на JavaScript. При завантаженні вона відображає дані про існуючі адреси нашої столиці лише на тих вулицях, де є об'єкти будівництва з проблемним станом. Це пояснюється наявністю інформації лише по місту Київ. Збільшений вигляд веб-інтерфейсу користувача:

Область	Населений пункт	Вулиця	+	-		
Київ ▾	м. Київ	алея Вернадського Академіка				
Додати будинок	Зберегти зміни в БД	Видалити з таблиці	Видалити з бази даних	Зберегти таблицю у CSV-файли	Генерувати випадкові дані	Спрогнозувати стан

Рисунок 2.3.1 – Інтерфейс користувача веб-проекту

У зв'язку із відсутністю даних про об'єкти житлового будівництва в інших областях та їх населених пунктах вибір варіанту із випадного списку «Область» не призведе до якихось змін. Наш проект має потенціал для роботи з багатьма містами, але для цього потрібні дані не тільки про нерухомість, а й про всі населені пункти, існуючі адреси. Однак, вибір вулиці можливий дуже зручним способом шляхом введення перших літер необхідної назви:

будівельних матеріалів. Додати новий пункт можна шляхом введення нової назви зліва від кнопки «+» та натисканням на неї, видалення обраного у випадному списку значення здійснює «-». Інші значення таблиці вводяться з клавіатури або обираються в списку.

Кнопка «Зберегти зміни в БД» здійснює збір повної інформації в таблиці, надсилає її за адресою «UpdateDatabase/» засобами Ajax у вигляді масиву. Відповідна функція Python циклічно формує та виконує SQL-запити, використовуючи модуль psycopg2 із вказаними параметрами. Виконання даної функції може призвести до недовготривалої затримки появи реакції на дії користувача при великій кількості даних в таблиці, що спостерігається при поточних параметрах веб-додатку і залежить від потужності процесора.

Кнопка «Видалити з таблиці» видаляє виділені пункти з інтерфейсу користувача. При необхідності можна видалити всі пункти, натиснувши на відповідне поле у заголовку першого стовпця таблиці. Схожу функцію виконує об'єкт «Видалити з бази даних», який спочатку збирає обрані рядки в масив, та за допомогою Ajax надсилає його середовищу Django за адресою «DeleteFromDatabase/». Відповідна функція циклічно обробляє масив, формуючи та виконуючи необхідні SQL-запити.

Для подальшого аналізу існуючих даних за допомогою інших засобів нами було передбачено відповідну функціональну можливість «Зберегти таблицю у CSV-файли». Вона полягає у створенні трьох файлів:

- поточної таблиці із стовбцями та даними веб-сторінки, назва цього файлу залежить від населеного пункту («Нерухомість у м. Київ.csv»);
- навчальних даних, які містять інформацію про завершені або затримані об'єкти будівництва, рік здачі в експлуатацію яких не перевищує 2022 (файл DataForLearning.csv);
- тестових даних, для яких потрібно отримати прогноз стану: до них включаються майбутні будинки із роком введення в експлуатацію не нижче 2023 (файл DataForPrediction.csv).

Таблиці із навчальною та тестовою інформацією, окрім вищенаведених стандартних стовбців, мають наступні: «Кількість недобудов на вулиці», «Кількість недобудов за рік», «Рейтинг матеріалів». Їх потреба обумовлена особливостями роботи інструментів для аналізу даних, які працюють лише з числами. Відомості про конкретний рік, вулицю, номер будинку, перелік будівельних матеріалів, висоту стелі тощо не допоможуть програмним інструментам виявити закономірність виникнення проблемних об'єктів будівництва без їх попередньої обробки та узагальнення. Тому ми ввели стовбці, які містять дані про частоту появи вказаних критеріїв. Ця інформація формується frontend-частиною проекту шляхом використання структури даних *map*, відомою як асоціативний масив або словник, і представляє собою пару «ключ – значення». Завдяки цьому вдається легко збільшувати кількість однакових об'єктів. Функція, яка виконує збір даних з таблиці, в нашому коді має назву *GetDataFromTable* і в якості аргументів приймає два масиви – для навчальних та тестових даних.

Через вищезазначений дефіцит достовірної інформації про успішно завершені об'єкти будівництва ми були вимушені передбачити можливість генерувати випадкові дані. На нашу думку, вони виходять максимально наближеними до реальних. Перед їх побудовою створюється перелік головних будівельних матеріалів, що зустрічаються на веб-порталі, з якого ми брали відомості про нерухомість в місті Київ. Ними є «цегла», «к/б панель», «цегла силікатна», «з/б панель», «газоблок», «утеплювач», «шлакоблок», «вентильований фасад», «керамблок», «монолітно-каркасний». Алгоритм працює таким чином, що будівельний об'єкт не може складатись лише з «утеплювача» та «вентильованого фасаду». Рік будівництва становить випадкове значення між 1967 та 2027. Стан «Будується» або «Проектується» отримують лише будівлі, термін здачі в експлуатацію яких знаходиться в майбутньому. Інші конструкції отримують помітку «Зданий». Оскільки відомості про нерухомість із проблемним станом є достовірними, існуючі об'єкти з порожніми полями не отримують статус «Невдала будівля». Отже, після

натискання на кнопку «Генерувати випадкові дані» ми отримуємо повністю заповнену таблицю, яка виглядає таким чином:

<input type="checkbox"/>	№	Вулиця	№ буд.	Рік	Матеріали	Кількість поверхів	Висота стелі	Стан	Прогноз
<input type="checkbox"/>	1	вул. Ракетна	10	2011	монолітн <input type="checkbox"/>	+	24	2.83	Зданий <input type="checkbox"/>
<input type="checkbox"/>	2	вул. Ракетна	11	1968	цегла сіл <input type="checkbox"/>	+	3	2.89	Зданий <input type="checkbox"/>
<input type="checkbox"/>	3	вул. Ракетна	12	2020	шлакобл <input type="checkbox"/>	+	13	3.18	Зданий <input type="checkbox"/>
<input type="checkbox"/>	4	вул. Ракетна	13	2010	монолітн <input type="checkbox"/>	+	19	2.64	Зданий <input type="checkbox"/>
<input type="checkbox"/>	5	вул. Ракетна	14	2020	монолітн <input type="checkbox"/>	+	1	3.09	Зданий <input type="checkbox"/>
<input type="checkbox"/>	6	вул. Ракетна	15	1994	цегла сіл <input type="checkbox"/>	+	20	2.94	Зданий <input type="checkbox"/>
<input type="checkbox"/>	7	вул. Ракетна	16	2003	утеплюв: <input type="checkbox"/>	+	11	2.97	Зданий <input type="checkbox"/>
<input type="checkbox"/>	8	вул. Ракетна	17	1969	утеплюв: <input type="checkbox"/>	+	18	2.70	Зданий <input type="checkbox"/>
<input type="checkbox"/>	9	вул. Ракетна	18	2021	монолітн <input type="checkbox"/>	+	21	2.58	Зданий <input type="checkbox"/>
<input type="checkbox"/>	10	вул. Ракетна	19	2017	монолітн <input type="checkbox"/>	+	19	2.43	Зданий <input type="checkbox"/>
<input type="checkbox"/>	11	вул. Ракетна	2	1962	вентильс <input type="checkbox"/>	+	21	2.98	Зданий <input type="checkbox"/>
<input type="checkbox"/>	12	вул. Ракетна	20	2024	вентильс <input type="checkbox"/>	+	2	2.48	Будується <input type="checkbox"/>

Рисунок 2.3.4 – Таблиця з випадковими даними про об’єкти нерухомості

Головною функціональною можливістю нашого проекту є прогнозування стану конструкцій, які знаходяться на етапі проектування, будівництва або на фінальній стадії здачі в експлуатацію. У frontend-частині проекту вона базується на використанні вищеописаної функції *GetDataFromTable*, яка збирає навчальні та тестові данні. При натисканні на відповідну кнопку засоби Аях передають на URL-адресу «predict/» відповідно два масиви рядків разом зі стовпцями. Після цього код на мові програмування Python перетворює ці дані в зручний для обробки вигляд, виконуючи процедуру *GetDataFrameFromLinesOfCSVFormat(lines)*. Результатом її роботи стає двовимірний масив Pandas DataFrame, назви колонок якого повністю ідентичні тим, які потрапляють у CSV-файл, окрім додаткового стовпця «Стан», в якому текстова інформація перетворена на числову (0 або 1).

Наступним кроком стає формування значень X та Y для навчання моделі. Параметр Y представляє собою одновимірний масив станів, X – вся інша інформація за виключенням зайвих стовпців, зокрема тих, які можуть включати в себе символи окрім цифр. Шляхом виконання команди *drop* ми позбуваємось таких полів як «Вулиця», «№ буд.», «Рік», «Матеріали», обидва «Стани» (числовий і текстовий). Отже, дані, необхідні для здійснення прогнозування отримуються наступним кодом:

```
y = DataForLearning.iloc[:,10].values
```

```
DataForLearning.drop(DataForLearning.columns[[0,2,3,5,9,10]],axis=1,inplace=True)
```

```
x = DataForLearning.iloc[:,0:]
```

```
DataForPrediction.drop(DataForPrediction.columns[[0,2,3,5,9,10]],axis=1,inplace=True)
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=0)
```

Ми бачимо, що виділення даних у відповідні змінні, а також видалення непотрібної інформації виконуються «черговим» способом. Після кількох експериментів ми зрозуміли, що ці команди не можуть бути частиною функції *GetDataFrameFromLinesOfCSVFormat(lines)*, оскільки ми взаємодіємо як з навчальними, так і з тестовими даними у строго заданій послідовності, обробляючи зовсім різні стовпці.

Отримавши необхідну інформацію після виконання *train_test_split* здійснюється виклик підпрограми *GetBestPredictions*, яка в якості аргументів додатково отримує масив *DataForPrediction*. Принцип її роботи полягає в отриманні прогнозів за допомогою трьох інструментів: логістична регресія (*LogisticRegression*), випадковий ліс (*RandomForestClassifier*), метод опорних векторів (*Support vector machine – SVM*) і повернення користувачу результатів з найбільшою точністю. Перелічені моделі детально аналізуються в наступних частинах нашої роботи. Після формування результатів аналізу у вигляді масиву значень 0 або 1 здійснюється перетворення числових величин на текстові («Успіх» або «Недобудова»), завантаження результатів у форматі JSON (JavaScript Object Notation – текстовий формат обміну даних), їх відображення на веб-сторінці лише для майбутніх об'єктів житлового будівництва.

2.4 Опис використаних моделей

У процесі створення продукту, який має виконувати вищезазвані задачі, ми маємо вирішувати задачі регресії та класифікації. Регресія – це форма зв'язку між випадковими величинами, закон зміни математичного сподівання однієї випадкової величини залежно від значень іншої. Розрізняють прямолінійну, криволінійну, ортогональну, параболічну та інші регресії, а також лінію і площину регресії. Класифікація об'єкта – це номер або найменування класу, що видається алгоритмом класифікації в результаті його застосування. У машинному

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

навчанні завдання класифікації вирішується, як правило, за допомогою методів штучної нейронної мережі при постановці експерименту в вигляді навчання з учителем.

Наша задача полягає в розробці засобів для отримання відповіді на чітке запитання: чи стане майбутній об'єкт будівництва успішним. Отже, в якості результату розроблений код має повернути нуль або одиницю. Проаналізувавши існуючі моделі машинного навчання, ми виділили три з них, які в найбільшій мірі підходять для вирішення нашої проблеми. Ними є логістична регресія, випадковий ліс та метод опорних векторів.

Логістична регресія – це статистичний регресійний метод, який застосовують у випадку, коли залежна змінна є двійковою, тобто може набувати тільки двох значень (0 або 1). Механізм її роботи полягає у виявленні зв'язку між декількома незалежними (так званими регресорами або предикторами) і залежними змінними. Поріг відсікання змінюється від 0 до 1, які є розрахунковими значеннями рівняння регресії та часто називаються рейтингами. Метод використовується у медицині, наприклад, для визначення чи є пухлина злоякісною, або доброякісною. Прикладами також можуть слугувати сортування електронних листів на «спам» або «не спам», оцінка ймовірності результатів певних подій, зокрема повернення або неповернення боргу, виконання або невиконання роботи вчасно тощо.

Випадковий ліс – це ансамблевий метод машинного навчання (той, що включає в себе декілька алгоритмів, які навчаються одночасно та виправляють помилки один одного), який придатний для класифікації, регресії та інших завдань. Він працює за допомогою побудови численних дерев прийняття рішень під час тренування моделі й продукує моду для класів або усереднений прогноз (регресія) побудованих дерев. Його недоліком є схильність до перенавчання. Класифікація об'єктів проводиться шляхом голосування: кожне дерево відносить об'єкт, який класифікується до одного з класів, і перемагає той клас, за який проголосувала найбільша кількість дерев. Оптимальне число дерев підбирається таким чином, щоб мінімізувати помилку класифікатора на тестовій вибірці. Отже,

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

основною функцією цього метода є розподілення на класи різноманітних даних, наприклад, зображень, назв і т. п.

Метод опорних векторів – це спосіб аналізу даних для класифікації та регресійного аналізу (визначення залежності однієї величини від іншої) за допомогою моделей з керованим навчанням (з використанням учителя) із зв'язаними алгоритмами навчання, які називаються опорно-векторними машинами. Цей метод відноситься до групи граничних методів, який визначає класи за допомогою меж областей. За допомогою даного методу розв'язуються задачі бінарної класифікації. В основі методу лежить поняття площини рішень (*plane*), яка розділяє об'єкти з різною класовою приналежністю. Метод відшукує зразки, які знаходяться на межах між двома класами, тобто опорні вектори. Класифікація вважається якісною, якщо область між межами порожня. Отже, цей метод зводить навчання класифікатора до задачі оптимізації, яка розв'язується евристичними алгоритмами (тими, що не є гарантовано точними або оптимальними, але вважаються достатніми для вирішення поставленого завдання). Він може застосовуватись для керування складними електромеханічними системами та нелінійними об'єктами, виконує функції спостерігача, ідентифікатора невідомих параметрів.

Бібліотека *sklearn* (*Scikit-learn*) для мови програмування Python має дуже зручні засоби для практичного використання описаних моделей. Вона містить всі алгоритми та інструменти, які потрібні для вирішення задач класифікації, регресії та кластеризації, надає доступ до інструментів оцінки швидкодії та точності застосованих моделей. Вона є дуже зручною у використанні, підходить для спеціалістів навіть з початковим рівнем підготовки. Практичне застосування передбачає розділення набору даних на навчальні та тестові, підгонку даних до моделі, безпосереднє виконання прогнозів та перевірка точності на тестовій вибірці. Такі відомі компанії як J.P. Morgan та Spotify тримають в своєму арсеналі функціональні можливості даної бібліотеки.

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		33

Висновок до розділу

Для вирішення задачі виявлення об'єктів будівництва з проблемним станом ми розробили веб-проект, залучивши найновіші технології. Зовнішня частина веб-додатку (frontend) базується на використанні стандартних можливостей мов розмітки, стильового оформлення та програмування (HTML, CSS та JavaScript). Обраний фреймворк Django, необхідний для реалізації серверної частини, є безкоштовно розповсюджуваним продуктом, що для нас є однією з необхідних вимог. Він надає можливість роботи з мовою програмування Python, перевагами якої є наявність зручних бібліотек для аналізу даних та машинного навчання, простота синтаксису, легкість опанування, можливість запуску коду на багатьох платформах без попередньої адаптації.

Велика увага була приділена інтерфейсу користувача: деякі поля мають засоби швидкого введення назв, зокрема, вулиці, надають можливості додавання, редагування та видалення даних. Через недосконалість реалізованих інструментів передбачені можливості збереження даних у відповідні файли для подальшого аналізу за допомогою сторонніх скриптів та програм.

Для виявлення об'єктів будівництва з проблемним станом ми використовували такі моделі машинного навчання, як логістична регресія, випадковий ліс, метод опорних векторів. В їх основі лежать різноманітні способи вирішення задач класифікації та регресії, принципи вибору оптимальних значень. Не дивлячись на те, що вони виконуються в ізолюваному середовищі, яке потребує додаткових програмних, а в деяких випадках і апаратних засобів, більшість з цих інструментів вирізняються високою продуктивністю, оскільки написані на швидкодіючій мові програмування C, а не на більш зручній Python.

Не дивлячись на відсутність зручних засобів отримання достовірної інформації про об'єкти будівництва в нашій країні, нами була вирішена складна проблема отримання даних для навчання та тестування моделей. Наш веб-проект включає в себе інструменти для ручного введення даних про будівельні об'єкти, однак дана функціональна можливість не спроможна спростити процес рутинного поповнення бази даних без використання таблиць у зручному для

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						34
Зм.	Арк.	№ докум.	Підпис	Дата		

обробки вигляді. Ми розуміємо, що база даних, як і сам веб-додаток, мають величезний простір для розвитку. При отриманні великої кількості достовірних даних проект має потенціал стати довідковим веб-сайтом, надаючи окрім загальнодоступних відомостей точні прогнози, які здатні врятувати покупців нерухомості та інвесторів, унеможливити реалізацію деяких шахрайських схем.

До недоліків розробленого веб-проекту відноситься відсутність можливості створювати профілі користувачів або організацій, які функціонували б за принципами вищеописаного порталу ЄДЕСБ. Це потребує більш глибокої досвідченості в області frontend, фундаментальних навичок роботи з backend-частиною, зокрема знань в галузі ідентифікації та аутентифікації. Наступним недоліком нашого проекту ми бачимо використання випадкових даних у зв'язку з неможливістю залучення великої кількості достовірної інформації про реальні завершені та незавершені об'єкти будівництва. Вони розміщені на веб-сайтах, до баз даних яких ми не маємо доступ, а ручне передрукування тисяч найменувань лише тільки для одного міста займе дуже велику кількість часу. Тому достовірна інформація використовується лише для недобудов Києва саме через їх відносно малу кількість. На жаль, ми не змогли отримати статистичні дані подібного вигляду по іншим містам нашої держави, оскільки органи влади, зазвичай, реєструють їх лише у вигляді декларацій на вищезазначеному ресурсі. Це створює певні перешкоди для збору даних про будь-які будівельні об'єкти.

У процесі розробки вказаних рішень ми вдосконалили навички роботи з мовами розмітки, стильового оформлення, програмування, навчились практично використовувати інструменти для аналізу даних та машинного навчання. Слід відмітити простоту синтаксису мови Python, наявність зручних бібліотек для неї, використання яких в більшості випадків не потребує серйозної підготовки.

Ми вважаємо, що наявність і легка доступність подібних сервісів в арсеналі будівельників, інвесторів та покупців житла разом із вдосконаленою законодавчою системою, більш широкою базою даних про будівельні об'єкти може спричинити позитивні зміни на ринку нерухомості.

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		35

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування вище описаного проекту включає в себе заходи по його розгортанню на локальній або видаленій машині, практичний запуск відповідних веб-сторінок засобами браузера, експерименти з кодом, використання засобів відлагодження тощо.

Розробка веб-проекту, пов'язаного з аналізом даних і машинним навчанням, часто потребує створення зовнішніх рішень. Ми успішно реалізували додаток для роботи з даними про об'єкти будівництва, в минулому розділі висвітлили його функціональність. Однак неможливо передбачити та реалізувати абсолютно всі інструменти для взаємодії з інформацією. Тому для більш глибокого аналізу як існуючих відомостей, так і якості використаних моделей машинного навчання нами було створено додаткове ПЗ. Воно представляє собою скрипт на мові програмування Python із залученням засобів для формування висновків у вигляді графіків та діаграм. Даний розділ також присвячений тестуванню рішень, реалізованих у веб-проекті.

3.1 Розгортання та тестування веб-проекту для роботи з нерухомістю

Для практичного застосування створеного додатку на локальній машині була написана інструкція користувача англійською мовою, яка включає в себе послідовність дій по встановленню необхідного ПЗ (Python та PostgreSQL), а також перелік інструкцій, які потрібно виконати засобами командного рядка. Вона має наступний вигляд:

1. встановити Python та модуль psycopg2 ("pip install psycopg2") для роботи з базою даних засобами мови Python;
2. встановити PostgreSQL, запустити SQL Shell (psql) та ввести в ньому наступні команди:
 - CREATE DATABASE realestate;
 - CREATE USER sqluser WITH PASSWORD '5353A201';
 - ALTER ROLE sqluser SET default_transaction_isolation TO 'read committed';
 - ALTER ROLE sqluser SET timezone TO 'UTC';

– GRANT postgres TO sqluser;

3. запустити файл «run script.bat», який знаходиться в директорії DataBase та почекати кілька хвилин, доки всі дані будуть завантажені в БД;

4. Створити середовище Python Django, уводячи такі інструкції в командному рядку Windows:

- cd /d d:\
- python -m venv PythonEnvironment
- PythonEnvironment\Scripts\python.exe -m pip install --upgrade pip
- PythonEnvironment\Scripts\activate.bat
- pip install django
- pip install psycpg2
- pip install pandas
- pip install numpy
- pip install -U scikit-learn
- pip install matplotlib
- pip install seaborn
- cd PythonEnvironment
- django-admin startproject realestate
- cd realestate
- python manage.py createsuperuser
- python manage.py startapp webapp

5. Скопіювати вміст папки DjangoFiles в директорію PythonEnvironment, при запиті замінити існуючі файли;

6. Виконати команду "python manage.py migrate" для фіксування змін.

Дана інструкція була успішно протестована під операційною системою Windows 7 з використанням Python 3.8.10 та PostgreSQL 15.3.2, має без проблем працювати на інших ОС родини Windows та, можливо, UNIX (MacOS, Linux). При наявності проблем із експлуатацією даного проекту є можливість використовувати вже налагоджене середовище Python Django, яке міститься у

відповідному архіві. При виконанні пунктів інструкції можна легко змінити назви директорій, пароль тощо, у деяких випадках відредагувавши необхідні файли.

Активация сервера здійснюється шляхом запуску файлу
«PythonEnvironment\realestate\run server.bat», результатом чого має бути вікно:

```
C:\Windows\system32\cmd.exe
watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 10, 2023 - 22:20:39
Django version 4.2.2, using settings 'realestate.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 3.1.1 – Робота серверної частини проекту відразу після старту

В процесі роботи воно може відображати дані про здійснені запити або помилки в такому вигляді:

```
C:\Windows\system32\cmd.exe
[10/Jun/2023 23:05:49] "POST /GetSeveralColumns/ HTTP/1.1" 200 12
[10/Jun/2023 23:05:49] "POST /GetSeveralColumns/ HTTP/1.1" 200 12
Not Found: /favicon.ico
[10/Jun/2023 23:05:49,864] - Broken pipe from ('127.0.0.1', 59423)
[10/Jun/2023 23:05:52] "GET / HTTP/1.1" 200 32933
[10/Jun/2023 23:05:52] "POST /GetOneColumn/ HTTP/1.1" 200 46
[10/Jun/2023 23:05:52] "POST /GetOneColumn/ HTTP/1.1" 200 196094
[10/Jun/2023 23:05:52] "POST /GetSeveralColumns/ HTTP/1.1" 200 7018
[10/Jun/2023 23:05:52] "POST /GetSeveralColumns/ HTTP/1.1" 200 3225
```

Рисунок 3.1.2 – Функціонування серверної частини в процесі роботи

Після успішного запуску сервера у веб-браузері потрібно перейти на адресу <http://127.0.0.1:8000/> (для зручності був створений ярлик *Open web page* у папці *PythonEnvironment*). Сторінка має наступний вигляд:

Область	Населений пункт	Вулиця		
Київ	м. Київ	алея Вернадського Академіка	+	-

Додати будинок	Зберегти зміни в БД	Видалити з таблиці	Видалити з бази даних	Зберегти таблицю у CSV-файли	Генерувати випадкові дані	Спрогнозувати стан
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

№	Вулиця	№ буд.	Рік	Матеріали	Кількість поверхів	Висота стелі	Стан	Прогноз
1	вул. Тихорецька	1						
2	вул. Тихорецька	10						
3	вул. Тихорецька	11						

Рисунок 3.1.3 – Веб-сторінка для роботи з базою даних житлових об'єктів

3.2 Зовнішній скрипт для аналізу даних про об'єкти будівництва

Для більш детального аналізу інформації, яку надає веб-проект, ми створили скрипт, який розміщується в директорії *BuildStatistics*. Він включає в себе файли коду (*script.py*), зручного запуску реалізованого рішення (*script.bat* для ОС Windows), тестові та навчальні дані у CSV-форматі. Результатами його виконання стають 19 графічних файлів формату *png*, а також дані прогнозування (*predictions.csv*), виводиться інформація про точність використаних моделей.

При роботі даного скрипта дані зчитуються з файлу і перетворюються на двовимірний масив із відповідними рядками та їх стовпцями шляхом виконання функції *GetArrayFromLinesOfCSVFormat*, яка в якості параметру отримує назву файлу. Після цього відбувається підготовка даних до аналізу: на відміну від веб-проекту, ми маємо зберігати не тільки об'єкти *Pandas DataFrame*, а й звичайні масиви (*Array*). Збір та збереження статистичної інформації виконує підпрограма *BuildPlotOfPredictions*, яка в якості параметрів отримує тестові та навчальні дані, заголовки на малюнках, назву вихідного файлу. Також вказується ім'я стовпця таблиці, інформацію в якому слід збільшити на зазначені кроки, представлені у вигляді переліку цілих або дробових чисел. У процесі своєї роботи скрипт редагує лише тестові дані (ті, для яких потрібно здійснити прогноз), оскільки ми виходимо з того, що інформація для навчання моделей є сталою та має бути отримана з достовірних джерел. Кількість ітерацій змін становить 10.

Щоб збільшити один стовбець, наприклад, на три різних кроки, ми повинні мати три однакові копії цього стовпця, та ітеративно збільшувати кожне значення в них. Оскільки дуже важко наочно проілюструвати велику кількість даних, ми обчислюємо середнє значення зміненого стовпця на кожному кроці його зміни. Після отримання даних здійснюється ще одне усереднення: кількість значень не має перевищувати кількість кроків, на які збільшувались числа у зазначеному стовпці. Далі виконується побудова графіка та стовпчастої діаграми із зазначеними розмірами шрифту та «полотна», точністю дробових чисел, надписами. Кожен малюнок має однакову структуру і складається із заголовку,

назви проаналізованого стовпця (розміщується вертикально), проценту недобудов (завжди знаходиться на горизонтальній осі), легенди із зазначеними кроками та кольорами відповідних ліній. Для зручності взаємодії з файлами імена графіків завжди починаються на «Plot», а діаграм – на «Barplot».

Було помічено, що інколи при побудові діаграми виникають труднощі, якщо розміри масивів X та Y різні. Для усунення цієї проблеми перші, в більшості випадків менші, значення видаляються командою `pop(0)`.

Для встановлення того, наскільки змінюється прогнозована частка недобудов, ми провели 5 експериментів над такими стовпцями: «Кількість недобудов на вулиці», «Кількість недобудов за рік», «Середній рейтинг матеріалів», «Середня кількість поверхів», «Середня висота стелі». Інші колонки не є повністю числовими або їх зовсім немає сенсу досліджувати шляхом редагувань (наприклад, «Рік»). Значення змінювались на одні й ті самі кроки (1, 2, 3). У результаті цього ми отримали малюнки із зовсім різним ступенем зрозумілості та інформативності. Вхідні дані були ретельно підібрані таким чином, щоб прогнози виявлялись як успішними, так і не успішними.

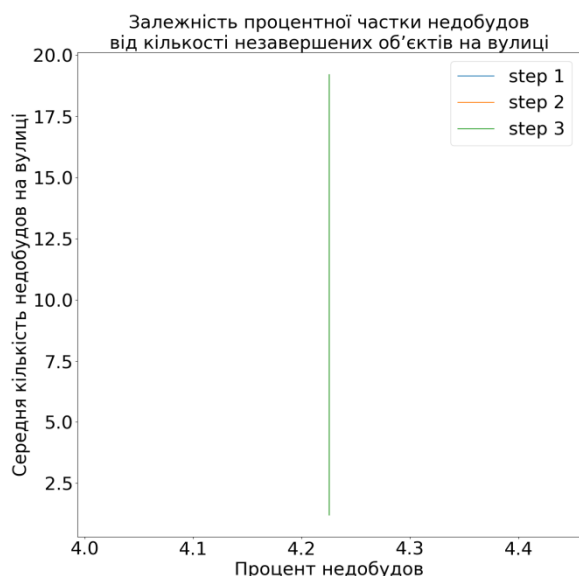


Рисунок 3.2.1 – Графік залежності процентної частки недобудов від «негативного рейтингу» вулиці

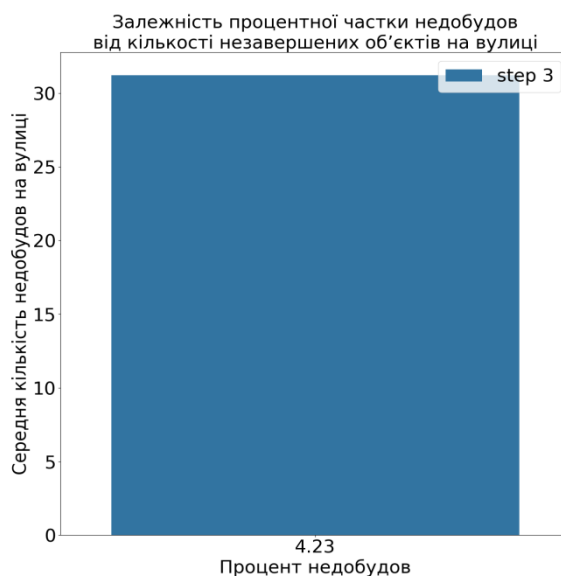


Рисунок 3.2.2 – Діаграма залежності процентної частки недобудов від «негативного рейтингу» вулиці

При побудові цього малюнка середні значення прогнозів для всіх трьох кроків різної величини виявились однаковими і становлять 4.23. Схожа картина спостерігається при експерименті із висотою стелі:

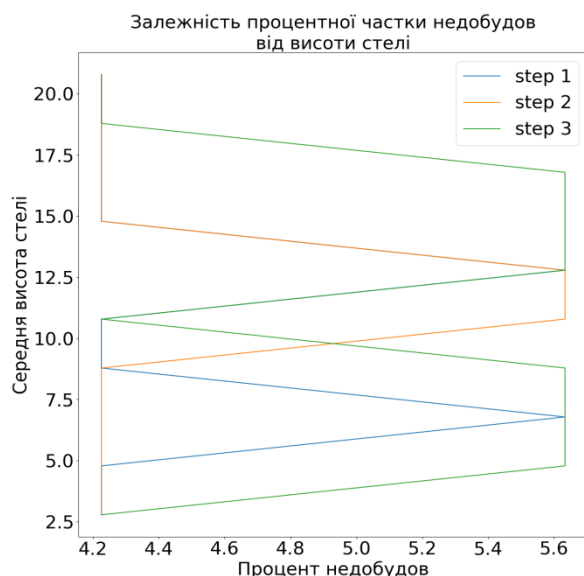


Рисунок 3.2.3 – Графік залежності процентної частки недобудов від висоти стелі

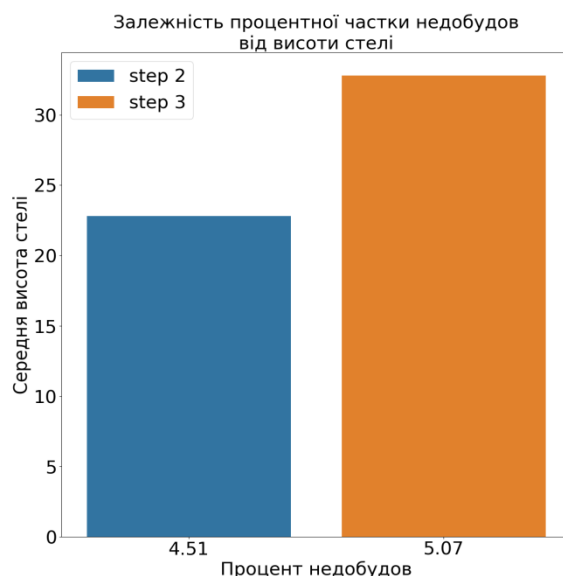


Рисунок 3.2.4 – Діаграма залежності процентної частки недобудов від висоти стелі

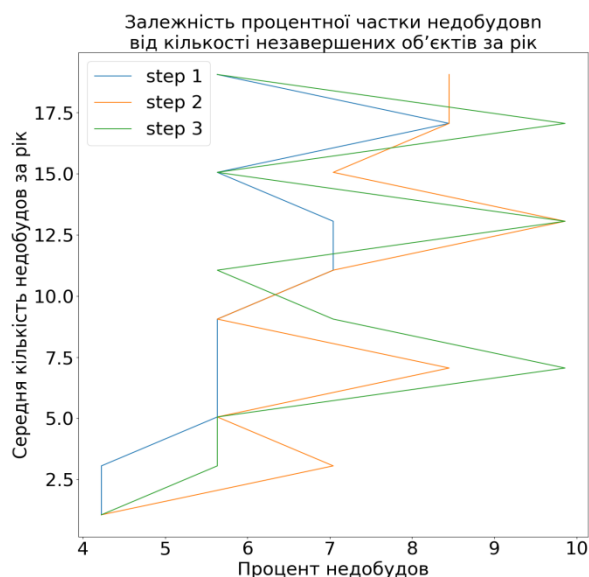


Рисунок 3.2.5 – Графік залежності процентної частки недобудов від збільшення «негативності» року

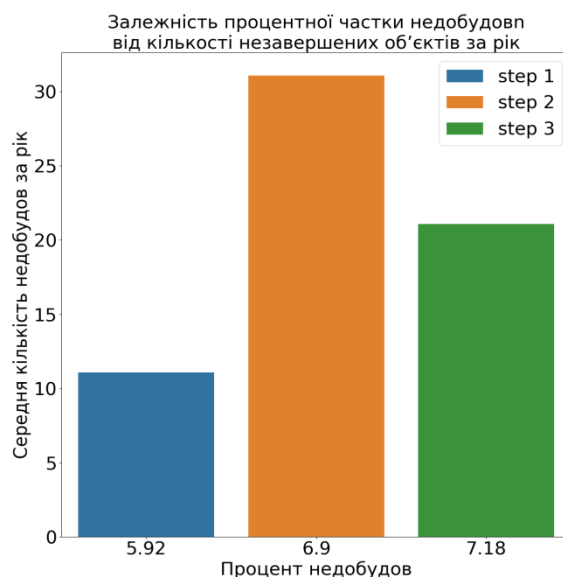


Рисунок 3.2.6 – Діаграма залежності процентної частки недобудов від збільшення «негативності» року

Ми бачимо, що збільшення кількості недобудов за вказані роки не закономірно впливає на прогнози. Це пояснюється недоцільністю дослідження даного параметра, оскільки аналізуються майбутні періоди часу, для яких не може бути отримана достовірна статистика про невдалі об'єкти будівництва.

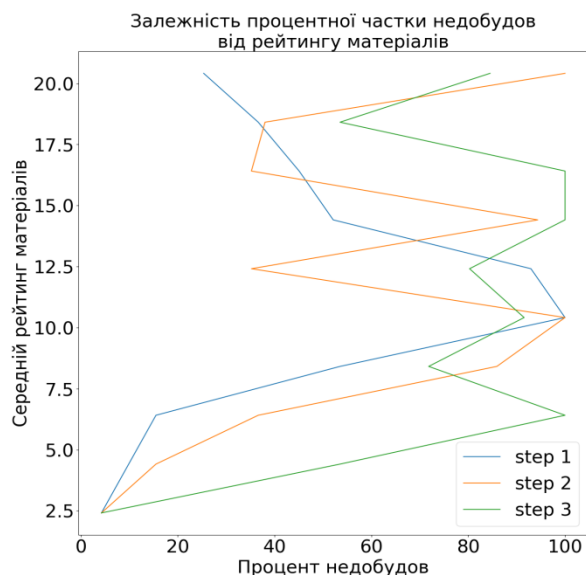


Рисунок 3.2.7 – Графік впливу рейтингу матеріалів

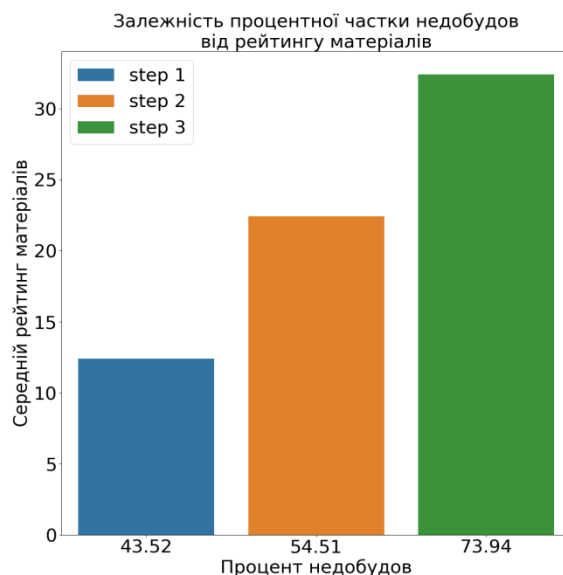


Рисунок 3.2.8 – Діаграма впливу рейтингу матеріалів

Це дослідження є більш осмисленим і здається правдоподібним. Ми бачимо, що збільшення рейтингу будівельних матеріалів, тобто використання кращих компонентів, призводить до зростанню кількості об'єктів будівництва з проблемним станом. Такий результат може мати місце в реальному світі, особливо в небагатих країнах, оскільки більш якісні матеріали, зазвичай, є дорожчими. Це призводить до збільшення витрат, а значить сприяє гіршому фінансуванню на етапі будівництва. В решті решт майбутнє житло може втратити постачальників коштів, та взагалі перетворитись на заморожену конструкцію. З іншого боку, збільшення рейтингу матеріалів може призвести до покращення ситуації на ринку завершеної нерухомості через те, що матеріали більшої якості можуть стати привабливішими для клієнтів та не завжди дорожчими, що збільшить імовірність вчасної здачі в експлуатацію. Тому при трактуванні подібних результатів слід враховувати реальну ситуацію в світі, може виникнути потреба змінити їх на протилежні.

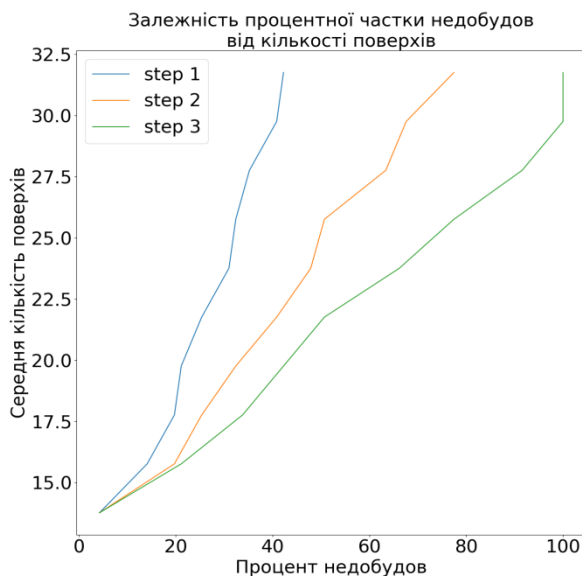


Рисунок 3.2.9 – Графік залежності процентної частки недобудов від кількості поверхів

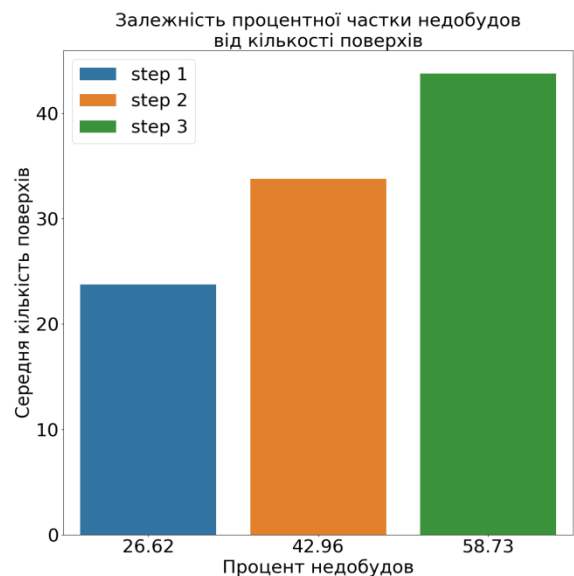


Рисунок 3.2.10 – Діаграма залежності процентної частки недобудов від кількості поверхів

Стрімке збільшення кількості поверхів, у порівнянні із завершеними об'єктами нерухомості, призводить до погіршення ситуації в будівельній галузі. Отже, ускладнення конструкції сприяє зменшенню ймовірності її здачі.

3.3 Оптимальний клас складності, верифікація окремих моделей

Окрім загальних експериментів, результатами яких ставали лише прогнози моделі з найбільшою точністю, ми приділили велику увагу дослідженню кожного окремого метода.

Деякі моделі машинного навчання, реалізовані у вигляді готових бібліотек для мови Python, мають параметр *sr*, який відповідає за складність. Основна його роль полягає в збереженні обчислювального часу шляхом відсічення розщеплень, які марно використовують апаратні ресурси. Вказуючи даний параметр, користувач повідомляє програмі, що будь-яке розподілення, яке покращує адаптацію за складністю, скоріш за все буде вилучене перехресною перевіркою, і тому програмі не треба його виконувати. Моделі, які базуються на дереві рішень мають схожий параметр *maxdepth*, який регулює максимальну глибину будь-якого вузла кінцевого дерева, причому корінний вузол вважається глибиною 0. У

процесі побудови проекту ми не побачили необхідність вказувати такі обмеження, щоб не погіршити точність роботи моделі. Проблема вибору оптимального класу складності стає актуальною при потребі в більшій продуктивності та при бажанні покращити її, не нехтуючи точністю.

Додаток ілюструє результати роботи моделей у вигляді кругових діаграм:

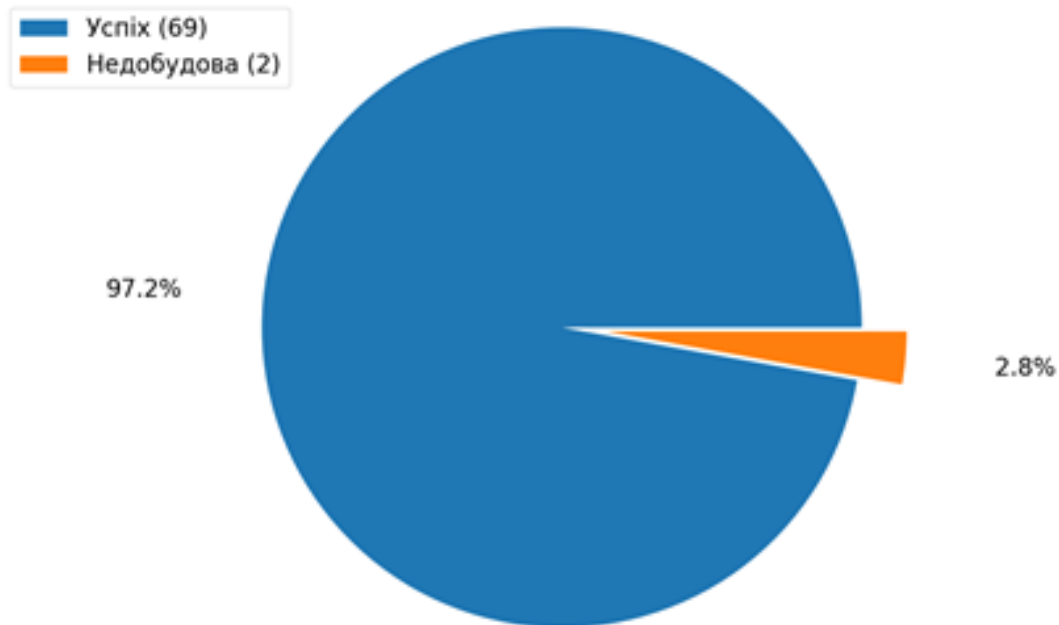


Рисунок 3.3.1 – Прогнозоване співвідношення зданих в експлуатацію будівель та недобудов (логістична регресія)

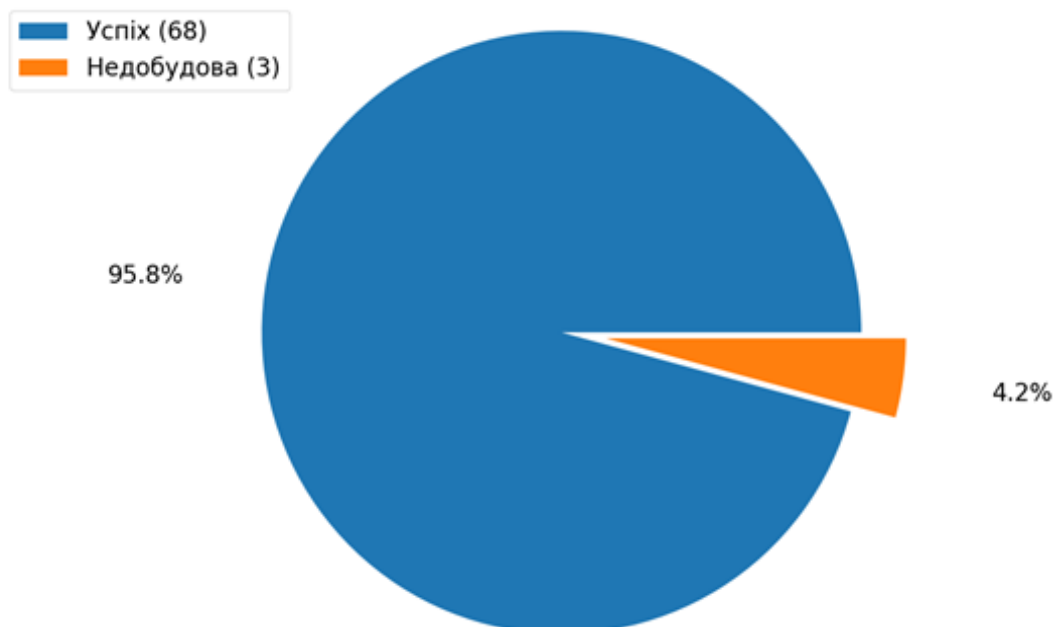


Рисунок 3.3.2 – Прогнозоване співвідношення зданих в експлуатацію будівель та недобудов (випадковий ліс)

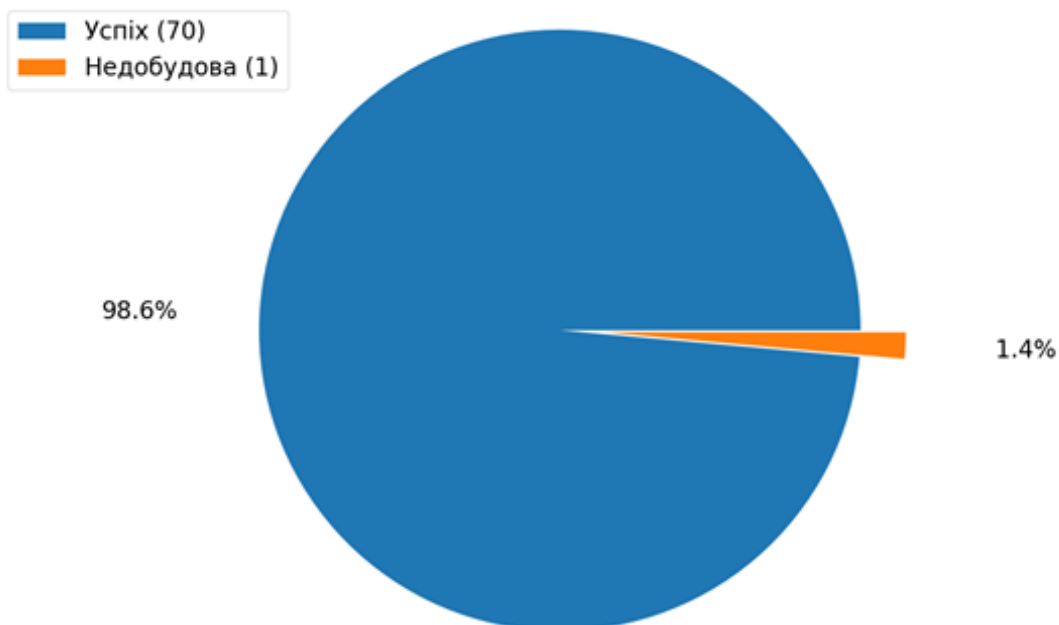


Рисунок 3.3.3 – Прогнозоване співвідношення зданих в експлуатацію будівель та недобудов (метод опорних векторів)

У ході проведення аналізу за допомогою трьох різних моделей ми помітили, що часто вони надають зовсім різні прогнози. Це пов'язано з принциповою відмінністю випробуваних нами алгоритмів. Також причинами таких розбіжностей може бути використання моделей із параметрами за замовчуванням: ми в жодному випадку не вказували додаткові уточнюючі та обмежувальні опції. Отримання повністю однакових результатів дослідження за допомогою різних моделей здається менш реальним. Було помічено, що в кількох випадках два алгоритми прогнозують однаковий негативний результат, що може в деякій мірі свідчити про достовірність результатів аналізу.

3.4 Графічне представлення навчальних даних

Не менш важливою справою є формування візуальних висновків про дані, які беруть участь в навчанні моделей. Через вже згадані причини вони були сформовані випадковим способом, та на перший погляд можуть здаватись цілком достовірними. Для їх побудови ми використовували структуру даних «словник», яка в коді оголошується фігурними дужками. Вона дозволяє зручно виконувати підрахунок кількості входжень кожного з параметрів. Отримана інформація

відображається у вигляді горизонтальних стовпчастих та кругових діаграм. Після певних редагувань ми отримали наступні зображення:

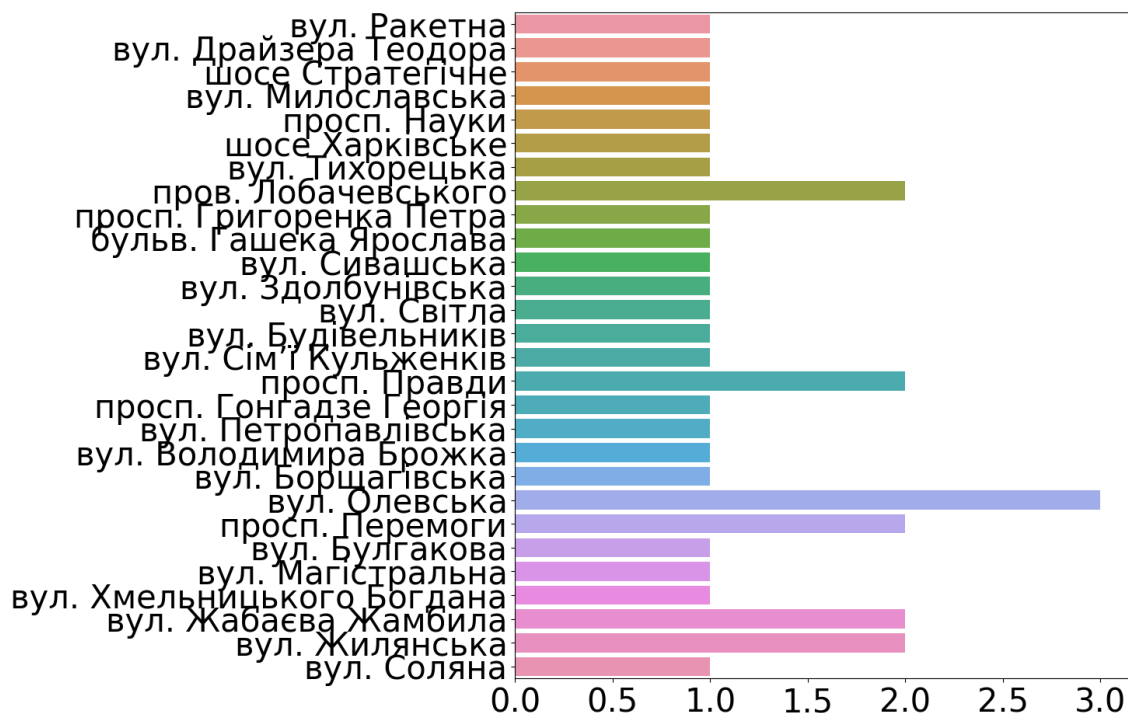


Рисунок 3.4.1 – Статистика недобудов за вулицями

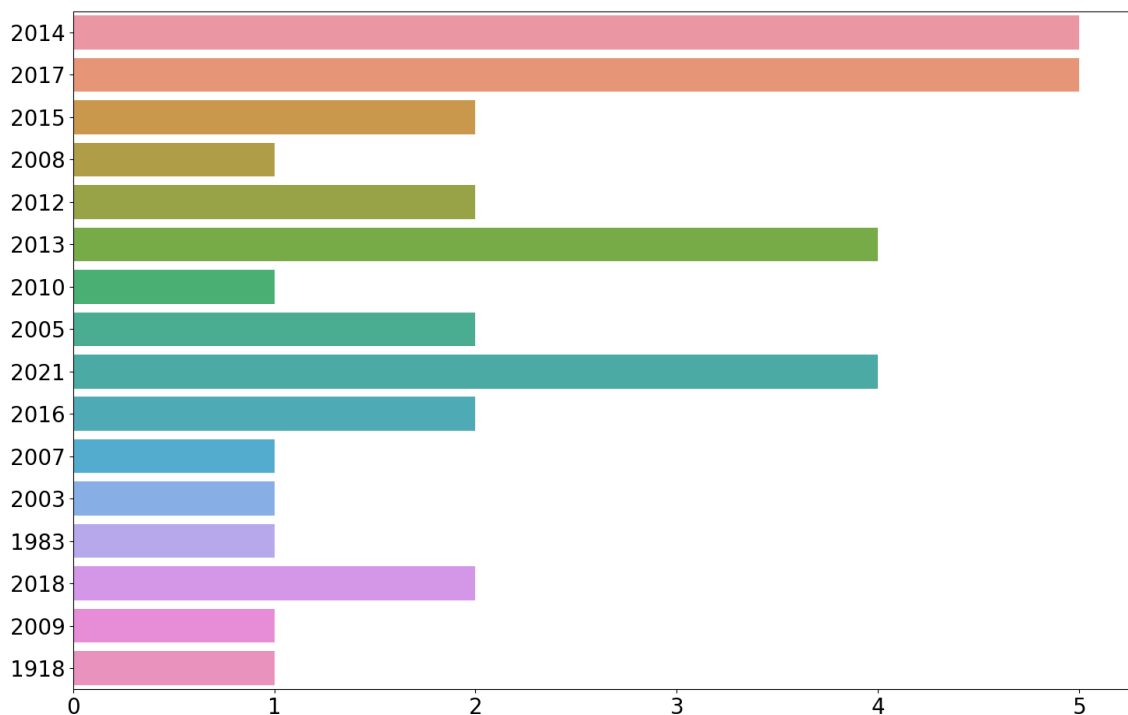


Рисунок 3.4.2 – Статистика недобудов за роками

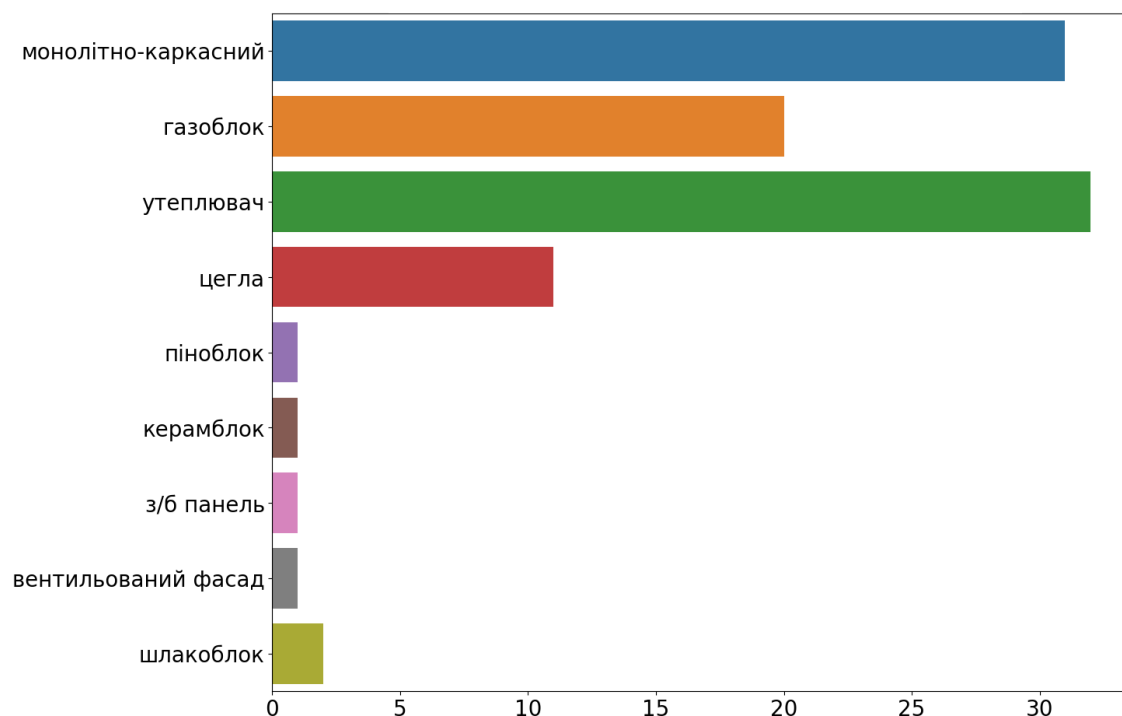


Рисунок 3.4.3 – Статистика недобудов за використаними матеріалами

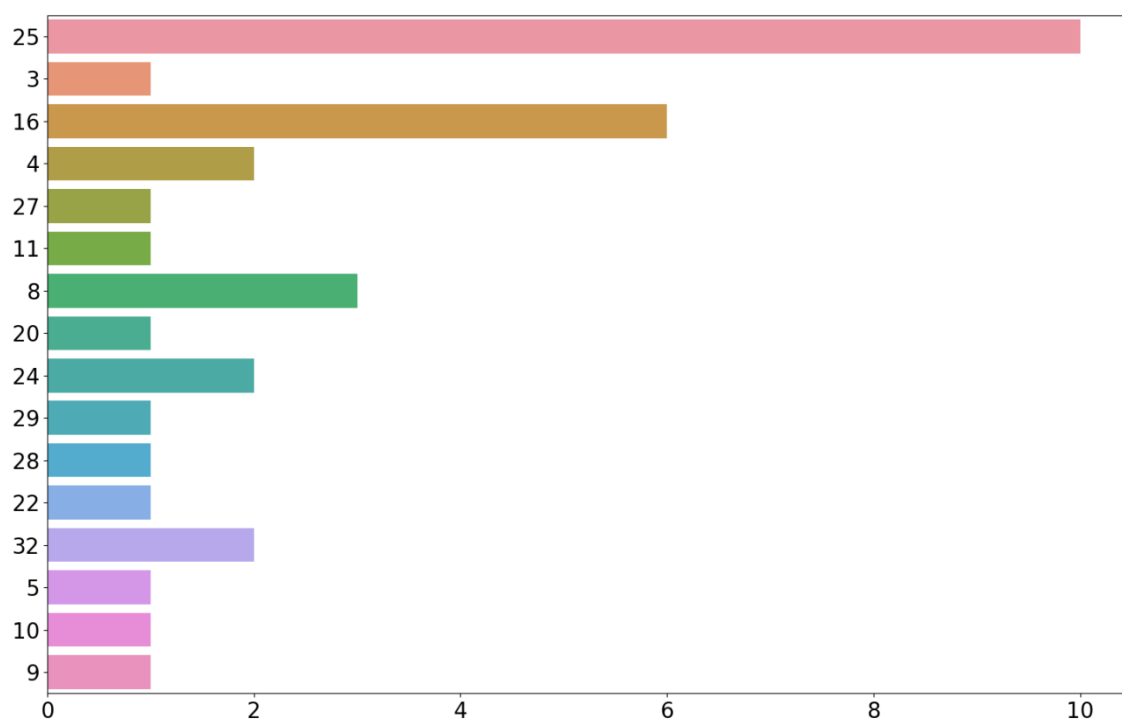


Рисунок 3.4.4 – Статистика недобудов за кількістю поверхів

Зм.	Арк.	№ докум.	Підпис	Дата

ЗПІ-зп01.080БАК.005 ПЗ

Арк.

47

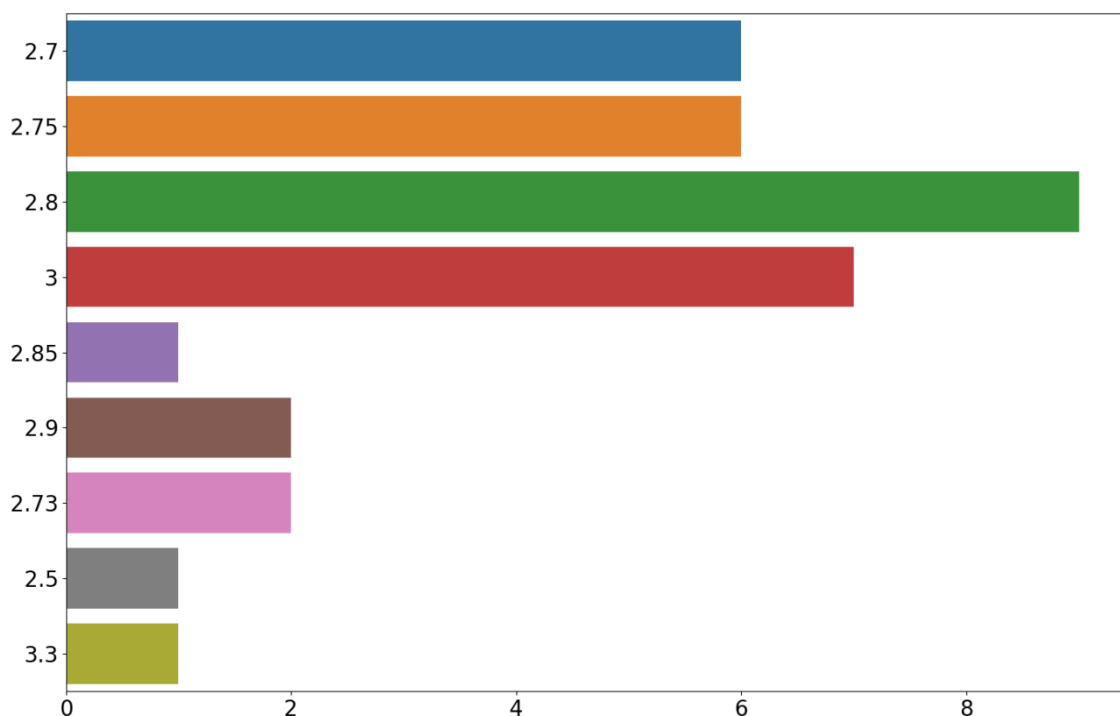


Рисунок 3.4.5 – Статистика недобудов за висотою стелі

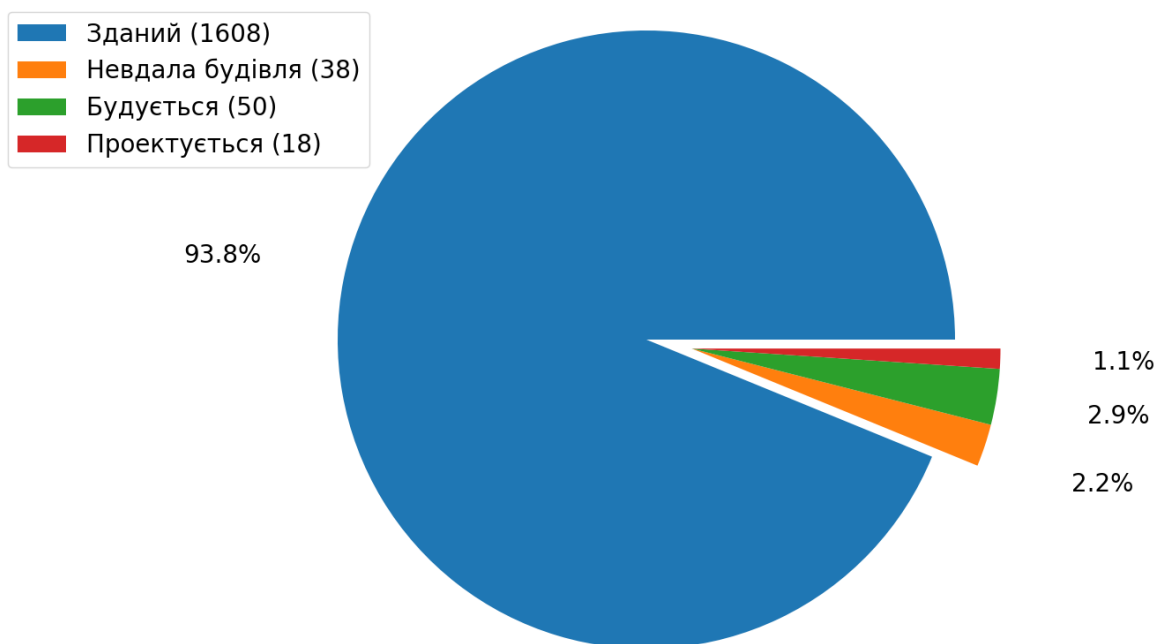


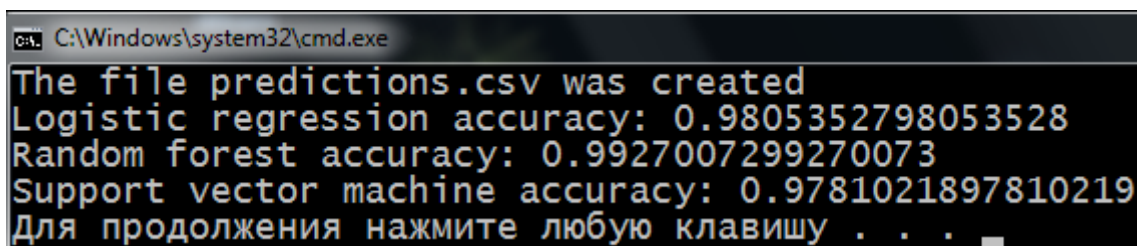
Рисунок 3.4.6 – Співвідношення різних стадій готовності об'єктів будівництва

Наведені дані є абсолютно достовірними та були отримані з веб-сайту Київської міської ради [6]. Ми бачимо, що найбільша кількість недобудов виникла в 2014 та 2017 роках. Переважна більшість вулиць не містить більше однієї недобудови, найгірший випадок – три об'єкта неподалік один від одного. Використання монолітно-каркасного та утеплювального матеріалів в найбільшій

мірі сприяє заморожуванню проектів. Найбільш невдалими є 25- та 16-поверхові конструкції. Найпроблемнішими розмірами стелі є 2.8 та 3 метри. Звісно, на ймовірність виникнення об'єктів будівництва з проблемним станом впливають не тільки вказані параметри, а й політичний, економічний та соціальний стан населеного пункту, фінансові спроможності інвесторів і т. п.

3.5 Оцінка якості реалізованих рішень

Головним показником якості результатів, отриманих засобами тієї чи іншої моделі машинного навчання, є точність – частка вірних відповідей в межах класу. Вона є співвідношенням об'єктів, що дійсно належать певному класу відносно всіх інших, які система віднесла до даного класу. Точність є інтуїтивно зрозумілою здатністю класифікатора не помічати вибірку як позитивну, якщо вона є негативною. Тому наш консольний додаток містить в собі можливість виведення оцінок точності реалізованих моделей:



```
C:\Windows\system32\cmd.exe
The file predictions.csv was created
Logistic regression accuracy: 0.9805352798053528
Random forest accuracy: 0.9927007299270073
Support vector machine accuracy: 0.9781021897810219
Для продовження натисніть будь-яку клавішу . . . _
```

Рисунок 3.5.1 – Оцінка точності моделей машинного навчання

Найбільш точним виявився випадковий ліс, який вважається універсальним, здатним вирішувати задачі класифікації, регресії, кластеризації, пошуку аномалій, селекції ознак тощо. Він є прикладом ансамблевого алгоритму, що ґрунтується на передбаченнях вирішальних дерев. Ідея ансамблевих моделей полягає у побудові великої кількості простих моделей, результати яких накопичуються. Інші моделі виявились достатньо точними та, на нашу думку, не потребують заходів для їх оптимізації. Такі показники обумовлені також невеликою кількістю ознак для аналізу, відносно великими розмірами навчальної збірки. Надмірна кількість вхідних параметрів для аналізу може негативно впливати на точність та швидкодію реалізованих моделей.

Висновок до розділу

Щоб перевірити роботу створених рішень, нами були розроблені зручні засоби для розгортання та запуску проекту. Вони включають в себе інструкцію по встановлення сервера на локальній машині, а також інструменти для запуску.

При виконанні даної частини роботи ми розробили скрипт, який здійснює всебічне тестування рішень, які лежать в основі веб-додатку. Користувач отримує результати у вигляді графіків, стовпчастих та кругових діаграм. Ми виконали загальну перевірку інструментів прогнозування шляхом зміни деяких параметрів на вказані кроки, узагальнили та усереднили отриману інформацію, сформували відповідні висновки, які базуються на результатах роботи моделей з найбільшою точністю. Була приділена увага дослідженню окремих методів машинного навчання шляхом надання відповідних статистичних відомостей. Встановлено, що досліджені моделі можуть давати різні результати на одних і тих самих даних.

Також ми візуально показали сталі дані, за допомогою яких навчаються моделі, сформульовано відповідні висновки. Опираючись на них, ми можемо дати рекомендацію не відноситись серйозно до статистики використання тих чи інших матеріалів в об'єктах незавершеного будівництва, при купівлі нерухомості не слід робити ставку на успішність вулиці в контексті відсутності на ній незавершених об'єктів. Приймаючи рішення по придбанню або фінансуванню майбутньої нерухомості, слід покладатись лише на власні фінансові можливості, загальну політичну та економічну ситуацію. Причина успішності чи неуспішності певного будівельного проекту може бути зовсім ніяк не пов'язаною із дослідженими параметрами. Її детальне встановлення та використання при прогнозуванні стану майбутніх конструкцій знаходиться за межами даного проекту і, можливо, потребує інших програмних засобів.

У процесі побудови описаного скрипту для тестування рішень ми покращили навички роботи з мовою програмування Python, опанували інструменти для побудови графіків та діаграм, розібрались з різноманітними параметрами візуалізації даних, навчились писати найбільш оптимальний код, позбавлений повторного використання інструкцій.

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						50
Зм.	Арк.	№ докум.	Підпис	Дата		

4 ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ ВІДНОВЛЕННЯ ОБ'ЄКТІВ БУДІВНИЦТВА З ПРОБЛЕМНИМ СТАНОМ

Одним із важливих кроків для зменшення кількості вже існуючих невдалих об'єктів будівництва є створення рішень для генерації хоча б приблизних результатів відновлювальної діяльності. Тому ми поставили перед собою ціль створити такий проект за допомогою мови програмування C++, вибір якої обумовлений наявністю навичок роботи з нею та здається нам найбільш доцільним у зв'язку із можливими потребами в максимальній продуктивності виконуваного коду. Користувачем мають задаватись перелік населених пунктів разом з відстанями між ними у відповідному текстовому файлі формату csv, мінімальні та максимальні кількості незавершених об'єктів будівництва в них, об'єм необхідних ресурсів для їх завершення, параметри появи необхідних матеріалів. Результатом виконання програми повинна стати таблиця у вигляді csv-файлу, яка містить приблизні результати будівельних заходів при зазначених даних. Додаток має працювати як в покроковому режимі з ілюстрацією дій в кожний найближчий момент часу, так і мати автоматичний спосіб генерації результатів, фіксуючі всі кроки в текстовому log-файлі.

4.1 Базові поняття

Для побудови якісних рішень ми маємо базуватись на поняттях, які лежать в основі дисципліни «моделювання систем». Воно є найбільш ефективним способом дослідження складних систем різного призначення, – технічних, економічних, екологічних, соціальних, інформаційних – як на етапі їх проектування, так і в процесі експлуатації.

Моделювання як спосіб пізнання використовувалось людиною з давніх часів. Але з появою комп'ютера моделювання систем збагатилось появою принципово нових методів моделювання таких, як імітаційне моделювання, еволюційне моделювання, методи групового урахування аргументів. Моделі і методи моделювання використовуються при створенні систем автоматизованого проектування, систем прийняття рішень, систем автоматизованого керування,

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		51

систем штучного інтелекту. Потрібність у розв'язанні задач моделювання систем виникає не тільки у науковця, але й у проектувальника, виробника, ділової людини під час повсякденної праці [19].

Сучасні технології моделювання не тільки полегшили і прискорили процес побудови та дослідження моделі, але й значно наблизили сприйняття інформації спеціаліста з моделювання систем і спеціаліста, що працює у галузі, яка моделюється. Результати моделювання, які представлені засобами 3D анімації, допомагають знайти спільну мову і розуміння між спеціалістами з моделювання систем та спеціалістами, що працюють у галузі, яка моделюється.

Отже, наші задачі мають вирішуватись шляхом імітації мережі масового обслуговування. Система масового обслуговування – це одне або декілька обслуговуючих пристроїв з чергою. Мережі масового обслуговування функціонують за наступними простими правилами. Вимоги, що надходять на обслуговування, проходять одну за одною СМО згідно указанного маршруту. На вході кожної СМО вимога намагається знайти один вільний пристрій з декількох паралельно функціонуючих. Правило, за яким вимога обирає той чи інший пристрій за умови, що вони вільні, може бути указано окремо у відповідності до процесу функціонування реальної системи. Якщо всі пристрої однакові, то вимога переглядає пристрої по порядку і займає перший, що виявився вільним. Якщо в момент надходження вимоги до СМО всі пристрої виявились зайнятими, то вимога при наявності черги у СМО намагається в неї потрапити, або залишає мережу масового обслуговування і вважається не обслугованою [19].

Черга у СМО обов'язково одна. Кількість місць у черзі може бути обмежена або необмежена. Якщо в реальному процесі не виникає проблеми з наявністю вільного місця у черзі, то при моделюванні така черга може вважатись необмеженою, навіть якщо місць у ній декілька. Отже, обмеження на кількість місць у черзі вводять, коли потрібно моделювати ситуацію «в даний момент вимога не може бути прийнята у СМО» [19].

Мережа масового обслуговування може мати такі ускладнюючі елементи, як декілька типів вимог, що обслуговуються, розгалуження та блокування маршрутів, черги із складними правилами упорядкування вимог у них і т. і.

Блокування маршрутів допомагають ввести в мережу МО елементи, що управляють процесом обслуговування. Наприклад, вилучити поламаний пристрій з процесу обслуговування, заборонити подальше просування вимоги до виконання деякої умови і т. і [19].

4.2 Опис реальної системи масового обслуговування

Перш ніж приступити до практичної реалізації проекту для моделювання процесів відновлення або добудови житлових об'єктів ми поставили перед собою задачу дослідження СМО на прикладі системи, яка може використовуватись в галузі реального виробництва. Для цього нами були поставлені наступні задачі:

1. реалізувати алгоритм імітації простої моделі обслуговування одним пристроєм з використанням об'єктно-орієнтованого підходу;
2. модифікувати алгоритм, додавши обчислення середнього завантаження пристрою;
3. створити модель за схемою, представленою на наступному зображенні:

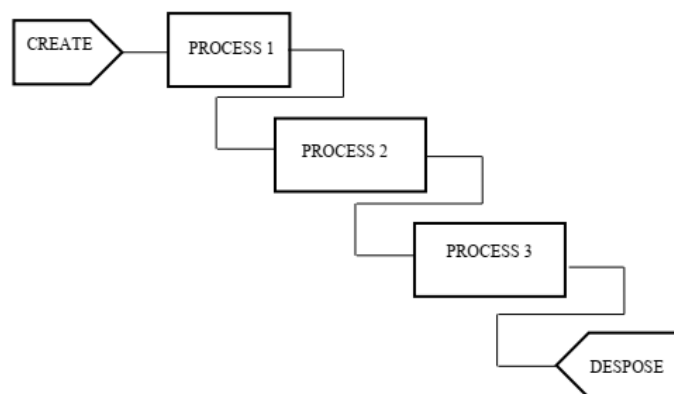


Рисунок 4.2.1 – Схема системи масового обслуговування

4. Виконати верифікацію моделі, змінюючи значення вхідних змінних та параметрів моделі. Навести результати верифікації у таблиці.
5. Модифікувати клас PROCESS, щоб можна було його використовувати для моделювання процесу обслуговування кількома ідентичними пристроями.

6. Модифікувати клас PROCESS, щоб можна було організовувати вихід в два і більше наступних блоків, в тому числі з поверненням у попередні блоки.

Щоб вирішити всі вищенаведені завдання нами був розроблений консольний додаток на мові програмування C++ з використанням об'єктно-орієнтованих засобів. Він представляє собою середовище для імітації мережі масового обслуговування, учасниками якої є *creator*, *robot* та *workshop*. Усі вони належать класу *device*, успадковуючи спільні властивості. Програма ілюструє роботу системи масового обслуговування, виводячи на екран стан кожного з «акторів» та перелік подій, які відбуваються в певний період модельного часу, що розраховується за принципом найближчої події. Головна ідея розробленого нами рішення полягає в тому, що *creator* створює завдання та передає їх об'єкту класу *robot*, який переміщується між виробничими приміщеннями (*workshop*), де ці завдання виконуються. Маршрути руху задані графом у вигляді матриці суміжності (зчитується з файлу), відстані та порядок слідування обчислюються за допомогою алгоритму Дейкстри. Усі «дійові особи», окрім *creator*, мають параметр «надійність», згідно з яким вони переходять в поламаний стан та відновлюють роботу через проміжки часу відповідного розміру. Максимальна кількість «аварій» також встановлюється, «актори» не приймають та не продовжують виконання завдань до «ремонткування». Таким чином об'єктом класу *robot* досягається необхідність переходу в попередні блоки.

Усі параметри роботи моделі задаються в програмному коді, зміни можна побачити лише після успішної компіляції. Для зручності роботи код було поділено на *cpp*-файли, назви методів та змінних знаходяться у *header*-файлах. *main.cpp* містить ініціалізуючі інструкції та запускає тести. *model.cpp* описує реалізацію моделі, а також зберігає результати у вигляді *csv*-файлів. *device.cpp* містить команди, які спільно виконуються об'єктами класів *creator*, *robot* та *workshop* (їх методи написані у відповідних файлах). *SharedFunctions.cpp* зберігає допоміжні підпрограми, які потрібні в будь-якій частині проекту.

Система може працювати як в автоматичному, так і в покроковому режимі. Вікно виглядає наступним чином:

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						54
Зм.	Арк.	№ докум.	Підпис	Дата		

```

C:\Windows\system32\cmd.exe
Experiment 1/51; changing parameter: mean delay (exponential distribution)
SIMULATION TIME: 89.81 /1000.00 ||||| 8.98 %
CREATOR | QUEUE | STORAGE | FAIL | PROGRESS | |||||
working 2/10 0/20 0 1/9 11.11% |||||
ROBOT | STATE | TASKS | FAIL | PROGRESS | | CURRENT LOCATION
00 waiting 1/11 0 28.50/28.50 | | Workshop-16
01 proc. 2 0/13 0 31.79/56.30 | ||||| route 3/4 3.29/13.00
02 waiting 1/14 0 14.80/14.80 | ||||| Workshop-17
03 proc. 2 1/13 0 9.87/40.50 | |||| route 1/3 9.87/13.70
04 waiting 2/9 0 13.60/13.60 | |||| Workshop-07
05 proc. 2 0/13 0 3.29/6.06 | ||||| route 1/1 3.29/6.06
06 proc. 1 12/13 0 92.31% | ||||| route
07 proc. 1 1/9 0 0.00% | ||||| route
08 waiting 0/0 0 0.00% | ||||| creator
09 waiting 0/0 0 0.00% | ||||| creator
WORKSHOP | STATE | QUEUE | FAIL | PROGRESS | WORKSHOP | STATE | QUEUE | FAIL | PROGRESS
01 waiting 0/3 0 | | 02 waiting 0/3 0 | |
03 waiting 0/4 0 | | 04 waiting 0/3 0 | |
05 waiting 0/2 0 | | 06 waiting 0/2 0 | |
07 proc. 3 1/4 0 ||| 08 waiting 0/2 0 | |
09 waiting 0/4 0 | | 10 waiting 0/4 0 | |
11 waiting 0/2 0 | | 12 waiting 0/4 0 | |
13 waiting 0/4 0 | | 14 waiting 0/2 0 | |
15 waiting 0/4 0 | | 16 proc. 3 1/4 0 |||||
17 proc. 3 1/4 0 ||||| 18 waiting 0/4 0 | |

Events of the time 89.85:
1. ROBOT 03: the total distance 13.16/40.50
along the route 1/3 Workshop-15->Workshop-16 (dist. 13.16/13.70) was gone

```

Рисунок 4.2.2 – Вікно системи масового обслуговування в покроковому режимі функціонування

Для коректності відображення слід встановити ширину 91 символ в налаштуваннях консолі, в майбутньому даний недолік буде усунений.

Creator генерує завдання у вигляді пари значень «номер цеху – тривалість виконання». Він має «відділи» для роботів, сховище для «макетів», а також перелік робочих приміщень. У першу чергу до роботів завантажуються раніше створені завдання, які вже знаходяться «на складі», лише у другу чергу створюються нові. Разом із завданнями об'єкти класу *robot* отримують значення інтервалу, після якого вони мають розпочати свої дії. Зрозуміло, що нові задачі потрапляють у сховище обмеженого розміру лише у випадку відсутності наявних роботів. При спробі додати завдання у переповнене сховище виникає помилка та значення змінної *CountOfFailures* збільшується. Процес генерації завдань запускається у заданий проміжок часу, який розраховується методом *GetDelay*. В залежності від заданих параметрів, він повертає момент часу наступної події за одним із трьох законів розподілу (рівномірного, нормального або експоненціального). Якщо закон розподілу заданий «некоректним» символом, інтервал наступної події встановлюється значенням вхідної змінної. Інші «актори» також використовують цей метод для обчислення моменту часу початку

подальших дій. Робота *creator* візуалізується відповідним індикатором виконання (progress bar) у вигляді символів вертикальної риски («|») і має наступний вигляд:

CREATOR	QUEUE	STORAGE	FAIL	PROGRESS						
working	2/10	0/20	0	1/9 11.11%						

Рисунок 4.2.3 – Функціонування модуля *creator*

Розмір часу, необхідний для генерації одного завдання, є випадковою величиною, яка знаходиться в заданому діапазоні. Складність завдання (максимальна кількість *workshop*, які має відвідати *robot*) також регулюється відповідними змінними. Вона не може бути меншою за 2, оскільки останнім пунктом завжди має бути сам *creator* (*robot* повертається до нього після виконання всіх завдань). Зрозуміло, що максимальне значення не може перевищувати кількість робочих приміщень. Отже, *creator* виконує перший етап «CREATE», зазначений на схемі, та вмикає подальші процеси.

Після отримання завдання у вигляді масиву структур «номер цеху – тривалість роботи» *robot* розпочинає «PROCESS 1». За нашим задумом, на реальному виробництві він є аналогом завантаженням каркасу майбутнього виробу на конвеєр, який переміщує деталі, поступово виконуючи їх збірку. У програмному коді цей процес реалізований шляхом відліку кількості цехів, які потрібно відвідати, із заданим інтервалом (змінна *TaskTime*). По його завершенню після певного проміжку часу, який може бути заданий за одним із законів розподілу, *robot* починає «PROCESS 2», який полягає в переміщенні макету до вказаного *workshop*. Цей процес включає в себе обчислення найкоротшої відстані, а також детальний вивід маршруту руху. Робота даного «актора» візуалізується:

ROBOT	STATE	TASKS	FAIL	PROGRESS		CURRENT LOCATION
00	waiting	1/11	0	28.50/28.50		workshop-16
01	proc. 2	0/13	0	31.79/56.30		route 3/4 3.29/13.00
02	waiting	1/14	0	14.80/14.80		workshop-17
03	proc. 2	1/13	0	9.87/40.50		route 1/3 9.87/13.70
04	waiting	2/9	0	13.60/13.60		workshop-07
05	proc. 2	0/13	0	3.29/6.06		route 1/1 3.29/6.06

Рисунок 4.2.4 – Функціонування модуля *robot*

Алгоритм *robot* також передбачає коректну поведінку при неможливості потрапити у цех (через відсутність вільного місця в черзі або поламаний стан *workshop*): він рухається до наступного виробничого приміщення, порушивши

заданий технологічний процес. У цьому випадку після першого обходу здійснюється перехід у попередні блоки, тобто *robot* намагається повторити маршрут, відвідавши не досягнуті *workshop*. Якщо це йому не вдається 3 рази, кількість аварій (*CountOfFailures*) збільшується, та цех видаляється з черги завдань. У випадку наявності лише одного «проблемного» приміщення роботом здійснюється циклічний рух між *creator* та цим *workshop* до тих пір, поки він або не потрапить до нього, або не видалиться.

У випадку успішного прийняття цехом, *robot* переходить у режим очікування сигналу про завершення обробки макету (час наступної події встановлюється максимальним). Після відвідування всіх виробничих приміщень або видалення недосяжних *robot* повертається у *creator* за новими завданнями, виконуючи процес «DISPOSE». Завершивши його, він не призначає час запуску *creator*, а просто очікує в черзі на нові завдання, які мають завантажитись зі сховища або згенеруватись згідно з розкладом. Кількість роботів також задається в програмному коді.

«PROCESS 3» виконується об'єктом класу *workshop*. У нашому випадку він полягає у відліку часу, отриманого від *robot*. Щоб уникнути надмірно велику кількість однотипних подій, ми ділимо цей час на 5 рівних частин, тобто інтерфейс користувача сповіщає про прогрес лише через інтервал $StepSize = WorkTime / 5.0$. У випадку виникнення аварії робота продовжується лише після ремонтування, що можна помітити у відповідному log-файлі:

```

36 651.09. WORKSHOP 06: the task 1/1 (robot 6, work time 22.92) was finished
37 671.82. WORKSHOP 06: the task 1/1 was executed for 20.00% (7.12/35.58)
38 678.94. WORKSHOP 06: the task 1/1 was executed for 40.00% (14.23/35.58)
39 683.47. Workshop 06: the state was changed to BROKEN
40 699.86. Workshop 06: the state was changed to active
41 706.97. WORKSHOP 06: the task 1/1 was executed for 60.00% (21.35/35.58)
42 714.09. WORKSHOP 06: the task 1/1 was executed for 80.00% (28.47/35.58)
43 721.21. WORKSHOP 06: the task 1/1 was executed for 100.00% (35.58/35.58)
44 728.32. WORKSHOP 06: the task 1/1 (robot 1, work time 35.58) was finished

```

Рисунок 4.2.5 – Фрагмент log-файлу, створеного компонентом *workshop*

Тут ми бачимо, що у час «678.94» було виконано 40% роботи, але у «683.47» *workshop 06* зламався, робота відновилась лише в «699.86», після цього було заново відраховано наступний проміжок часу. З частини цього log-файлу ми

також бачимо, що на повне виконання одного завдання *workshop* витрачає 120% модельного часу, оскільки останні 20% йому необхідні для переходу в попередній стан. Цей процес також включає в себе оновлення індикаторів, які мають наступний вигляд:

WORKSHOP	STATE	QUEUE	FAIL	PROGRESS
02	waiting	0/3	1	
04	waiting	0/4	3	
06	waiting	0/3	0	
08	proc. 3	1/4	2	
10	waiting	0/2	2	
12	waiting	0/2	0	
14	proc. 3	1/2	0	

Рисунок 4.2.6 – Вихідні дані про роботу *workshop*

Значення стовпця «FAIL» збільшується, якщо *robot* не зміг потрапити до нього через відсутність місця в черзі або через неробочий стан. Після обслуговування всіх *robot* він переходить в режим очікування, встановивши час наступної події рівним максимальному значенню типу *double*.

Як було помічено вище, перелік усіх подій зберігається у відповідних файлах, тобто кожен об'єкт класу *device* «веде» власний «щоденник», крім них є і AllEvents.txt, де перелічені абсолютно всі події, які відбулись у системі. Окрім log-файлів система створює і csv-файли, наповнені статистичними даними. Кожен з «акторів» має свої власні стовпці таблиць у відповідному файлі. Так, для *creator* у вікні Excel таблиця виглядає таким чином:

	A	B	C	D	E	F	G	H
1	Changing parameter: mean delay (exponential distribution)							
2	Mean delay	Max. delay	Size of	Avg.	Avg. task	Avg. time	Avg.	Count of
3	or min. delay	or deviation	storage	delay	complexity	for one task	load %	failures
4	27.68	0	25	30.67	9.02	6.19	27.39	44
5	52.25	0	25	34.3	10.12	6.38	27.2	50

Рисунок 4.2.7 – Статистичні дані про результати роботи компонента *creator*

У ній ми «реєструємо» середню або мінімальну затримку, максимальну затримку або відхилення (необхідна лише для генерації значень за рівномірним або нормальним законом розподілу), максимальний розмір сховища. У процесі роботи обчислюються середня затримка, середня складність створених завдань, середній час на генерацію одного завдання, середнє навантаження (процент

активного часу), кількість «аварій» через відсутність місця для нових макетів. Таблиці будуються відповідно до характеру експериментів один з яких містить в собі 10 або 11 «ітерацій», протягом них фіксуються зміни стандартно заданих значень і отримані результати.

4.3 Тестування розробленої мережі

Ми провели 5 експериментів, отримавши 5 таблиць з результатами роботи кожного учасника мережі. Перший з них показує вплив середньої затримки, яка обчислюється за експоненціальним законом розподілу [21]. Для *creator* графік має наступний вигляд:

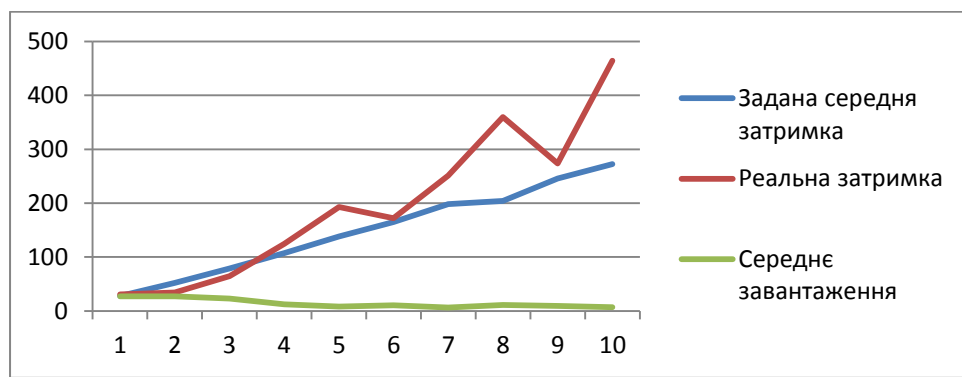


Рисунок 4.3.1 – Дослідження затримки модуля *creator*

Ми бачимо, що збільшення заданої користувачем затримки призводить до збільшення часу реальної затримки, але з певними перепадами, середнє навантаження (процент активного часу) при цьому зменшується. Ми робимо висновок, що *creator* коректно реагує на зміну вказаних параметрів, відхилення від норми можна пояснити постійним використанням випадкових значень.

Побудуємо подібний графік для *robot*:

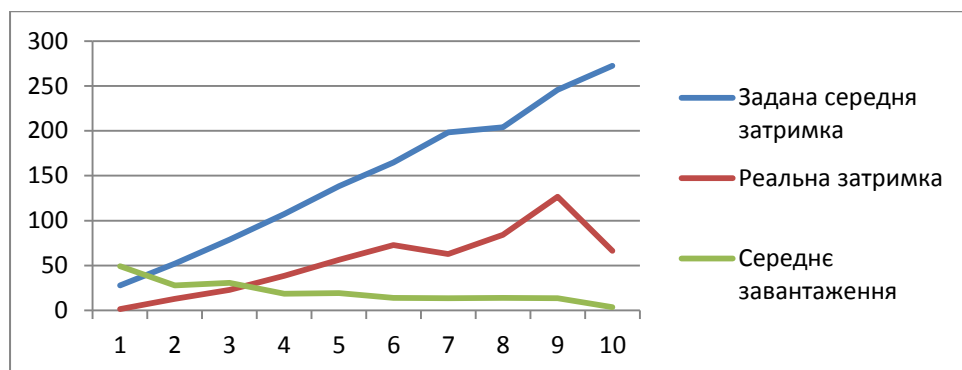


Рисунок 4.3.2 – Дослідження затримки модуля *robot*

Спостерігається подібна тенденція, що може свідчити про правильність реалізованих алгоритмів. Графік для *workshop*:

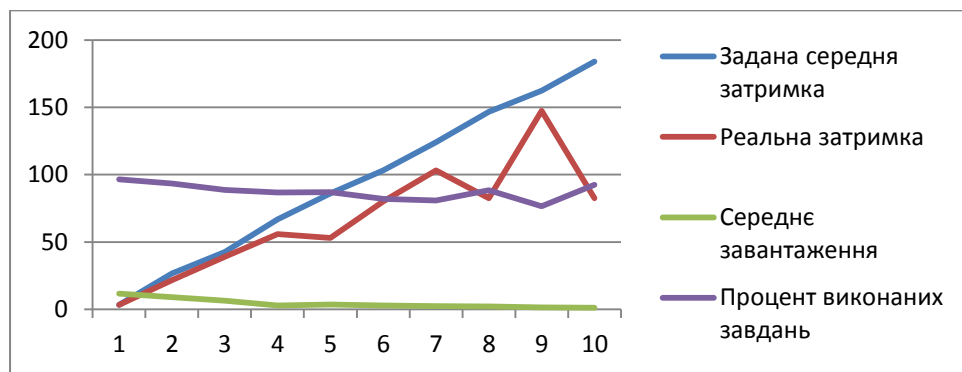


Рисунок 4.3.3 – Дослідження затримки і продуктивності модуля *workshop*

Тут, окрім вищезазначених параметрів ми проаналізували процент виконаних завдань. Було помічено, що він не завжди залежить від заданого параметру затримки, оскільки на нього впливають кілька факторів, включаючи надійність та швидкість *robot*. Розрахунки даного параметра за іншими законами розподілів призвели до майже тих самих результатів. При фіксованих значеннях графіки мають схожий вигляд.

Також нами було проаналізовано вплив однакової складності завдань на продуктивність та активність компонентів системи.

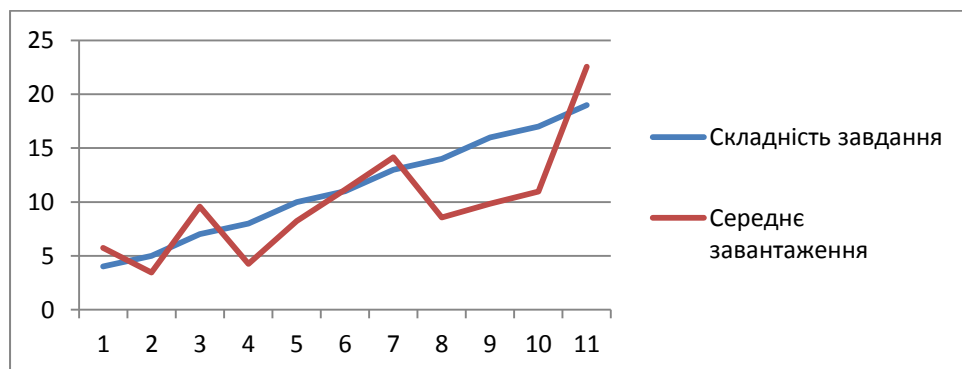


Рисунок 4.3.4 – Вплив складності завдання на середнє завантаження компонента *creator*

Ми бачимо, що збільшення значень даного параметру не завжди відповідним чином впливає на завантаженість пристрою, однак загальна тенденція має зростаючий характер.

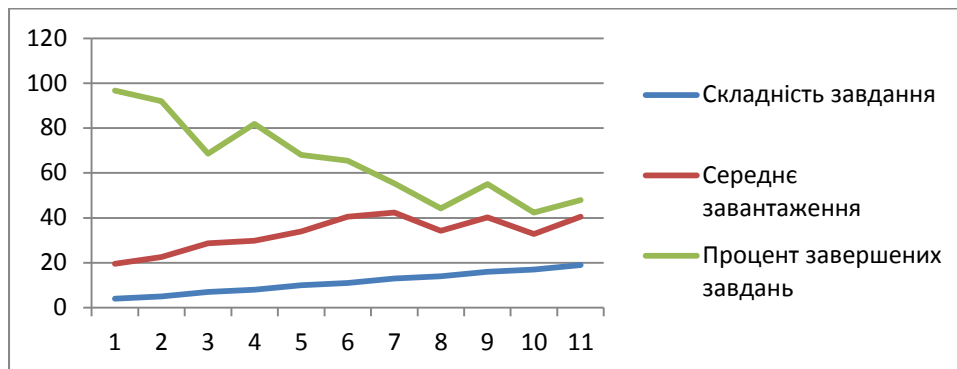


Рисунок 4.3.5 – Вплив складності завдання на середнє завантаження та продуктивність компонента *robot*

Тут видно, що разом зі зростаючою складністю завдання у більшості випадків зростає і розмір активного часу, а процент виконаних не завжди залежить від значення змінюваного параметра, хоча тенденція є низхідною.

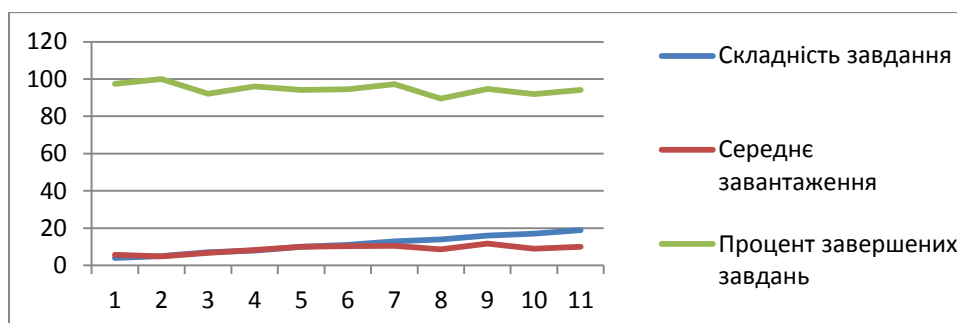


Рисунок 4.3.6 – Вплив складності завдання на середнє завантаження та продуктивність компонента *workshop*

На цьому графіку видно, що середнє завантаження та процент виконаних завдань не залежить від їх складності. Тобто *workshop* у найменшій мірі підвладний впливу *creator*, що не можна стверджувати про *robot*.

Також нами було проаналізовано середній та максимальний розміри черги для *workshop*. Було встановлено, що при заданих параметрах системи черга майже не використовуються, після виконання всіх експериментів значення цього параметру не перевищувало одиницю.

4.4 Система прогнозування результатів відновлення об'єктів будівництва з проблемним станом

Дослідивши роботу мережі масового обслуговування, ми зрозуміли, що описаний підхід може бути використаний для створення ПЗ, яке виконує прогнозування результатів відновлення нерухомості з проблемним станом.

Даний проект знаходиться на етапі розробки, проте його структура у вигляді header- та сpp-файлів, конструкторів усіх об'єктів, код для завантаження вхідних даних та виведення результатів вже сформовані.

```

SIMULATION TIME: 0.00 /1095.00 0.00 %
MANAGER CAR STORAGE FAIL FUNDS $ BNS.
10/10 0.00% 0 0.0000/1.0000
CARS STATE TASKS DELIVERED STUFF CURRENT LOCATION
00 waiting 0/0 0.00/0.00 0% Manager
01 waiting 0/0 0.00/0.00 0% Manager
02 waiting 0/0 0.00/0.00 0% Manager
03 waiting 0/0 0.00/0.00 0% Manager
04 waiting 0/0 0.00/0.00 0% Manager
05 waiting 0/0 0.00/0.00 0% Manager
06 waiting 0/0 0.00/0.00 0% Manager
07 waiting 0/0 0.00/0.00 0% Manager
08 waiting 0/0 0.00/0.00 0% Manager
09 waiting 0/0 0.00/0.00 0% Manager
LOCATION STATE BLDGS USED STUFF ($ BNS.) LOCATION STATE BLDGS USED STUFF ($ BNS.)
Bil..kva waiting 0/5 0.00/0.00 0 Che..hiv waiting 0/6 0.00/0.00 0
Dnipro waiting 0/34 0.00/0.00 0 Kharkiv waiting 0/29 0.00/0.00 0
Kherson waiting 0/11 0.00/0.00 0 Kre..huk waiting 0/8 0.00/0.00 0
Kry..Rih waiting 0/35 0.00/0.00 0 Kyiv waiting 0/70 0.00/0.00 0
Lviv waiting 0/12 0.00/0.00 0 Mykolaiv waiting 0/21 0.00/0.00 0
Nov..vka waiting 0/18 0.00/0.00 0 Ochakiv waiting 0/1 0.00/0.00 0
Odessa waiting 0/13 0.00/0.00 0 Poltava waiting 0/8 0.00/0.00 0
Sumy waiting 0/7 0.00/0.00 0 Uman waiting 0/3 0.00/0.00 0
Vin..sia waiting 0/9 0.00/0.00 0 Yuzhne waiting 0/0 0.00/0.00 0
Yuz..nsk waiting 0/2 0.00/0.00 0 Zap..hia waiting 0/27 0.00/0.00 0
  
```

Рисунок 4.4.1 – Програма для прогнозування результатів відновлення об'єктів будівництва з проблемним станом

Цей додаток зчитує дані про наявність шляхів та відстані між населеними пунктами. Оскільки дані про проблемні об'єкти будівництва відсутні для більшості міст, ми формували їх на основі достовірної інформації про Київ, що відбувається шляхом визначення процентного співвідношення між площею столиці та іншого меншого міста. Змінна, яка зберігає кількість проблемних об'єктів, отримує різницю між реальною кількістю недобудов у Києві та процентом площі від нього.

Роботою системи керує об'єкт класу *manager*. При його створенні користувачем задаються інтервали виклику (випадковий проміжок модельного часу в заданих діапазонах або закон розподілу), розмір коштів, які мають бути виділені протягом заданої кількості років, число транспортних засобів для доставки будівельних матеріалів у необхідні міста. Головними функціями менеджера є поетапна генерація грошової суми такого розміру, щоб за заданий період часу були виділені всі кошти. У зв'язку із наявністю дуже великої

кількості будівельних матеріалів за різноманітними цінами ми оперували лише їх вартістю: значення ваги або об'єму матеріалів представлено грошовим еквівалентом, що в більшості випадків не відповідає реальній дійсності (одна й та сама кількість різних матеріалів може мати зовсім різну ціну).

Після виділення коштів вони розподіляються між транспортними засобами (клас *car*), об'єм яких задається грошовою сумою. У випадку відсутності достатньої кількості транспортних засобів залишок потрапляє у сховище обмеженого розміру. При його переповненні виникає помилка, та ці гроші мають бути виділені пізніше. Транспортний засіб через заданий проміжок часу починає рух до міста, яке реалізовано у вигляді об'єкта класу *city*. Маршрут руху визначається алгоритмом Дейкстри.

Після потрапляння грошової суми до міста вона розподіляється між об'єктами класу *construction*, які містять інформацію про кількість коштів, необхідних для відновлення. Через задані проміжки часу об'єктом *city* виконуються «кроки» будівництва: на будівельних об'єктах витрачається певна кількість коштів. Користувач має можливість слідкувати за процесом у покроковому режимі. Після симуляції ми отримуємо log- та csv-файли, які містять дані про кількість виділених, доставлених та використаних коштів. Змінюючи параметри системи циклічним чином ми зможемо отримати різні прогнози результатів відновлення проблемних об'єктів будівництва та побудувати відповідні графіки, як це було виконано при реалізації класичної моделі масового обслуговування.

На початковій стадії розробки даного проекту була вирішена проблема ширини вікна консольного додатку, яка присутня у вищеописаній програмі для імітації СМО. Користувач може не піклуватись про налаштування командного рядка Windows, оскільки всі необхідні параметри задаються відразу після запуску. Це досягається шляхом використання можливостей Windows API. Недолік даного підходу полягає у відсутності можливостей запускати цей код на машинах під управлінням інших ОС.

Висновок до розділу

Окрім розробки інструментів для аналізу даних про житлові об'єкти ми поставили перед собою ціль створити ПЗ, яке може імітувати процес відновлення об'єктів будівництва з проблемним станом. Для цього ми вирішили використати систему масового обслуговування. Було розроблено програмне рішення для імітації роботи відповідної мережі, яке формує результати у вигляді файлів, придатних для аналізу сторонніми засобами, наприклад, Microsoft Excel. В його основі лежить принцип найближчої події.

Дійовими особами системи є: об'єкт *creator*, який генерує нові завдання; екземпляри класу *robot*, кількість яких задає користувач та функцією яких є доставка завдань у виробничі приміщення; тип *workshop*, представлений у вигляді зв'язного графу, об'єкти якого виконують отримані завдання. Роботу системи регулює певна кількість параметрів, до яких відносяться дані про часові затримки, максимальний розмір черги, складність завдань, відомості про надійність кожного з пристроїв тощо.

Побудувавши відповідні графіки, ми дійшли висновку, що зміна вказаних параметрів коректно впливає на інші показники системи. Проте розроблена нами програма не позбавлена недоліків. Наприклад, «актори» переходять у режим «аварії» навіть якщо знаходяться в стані спокою: у реальному світі таке дуже рідко трапляється. Тільки в деяких виключних випадках, зазвичай, під впливом зовнішнього середовища, пристрій може зламатись, знаходячись у неактивному стані. Для усунення цього недоліку нам слід змінювати стан пристрою лише в активний час, а не за весь період моделювання.

ПЗ для прогнозування результатів будівельної діяльності базується на тій самій мережі масового обслуговування. Воно має схожу структуру і представлено у вигляді взаємодії об'єктів класів *manager* (генерує і розподіляє будівельні матеріали, оперуючи грошовою величиною), *car* (здійснює доставку необхідних матеріалів на об'єкти будівництва), *city* (представляє собою міста, зв'язані шляхами із заданими відстанями, взаємодіють з об'єктами класу *construction*, контролюючи процес їх відновлення).

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						64
Зм.	Арк.	№ докум.	Підпис	Дата		

Ця програма знаходиться на початковій стадії розробки, проте вже реалізовано консольний інтерфейс користувача, а також засоби моніторингу результатів на кожному кроці моделювання. Вихідна інформація представляється у вигляді csv-файлів для подальшого аналізу сторонніми засобами. Нами використовувались певні засоби для встановлення таких параметрів відображення інформації на екрані, щоб, не залежно від поточних параметрів командного рядка, користувач бачив дані в коректному вигляді.

До недоліків практичного застосування спроектованих рішень відноситься наближеність отриманих результатів роботи кожного з компонентів. Часові інтервали та значення інших параметри в більшості випадків встановлюються випадковим чином, що може не відповідати закономірності функціонування об'єктів у реальному світі. Тому серйозні рішення не можуть базуватись на результатах імітації роботи описаних систем масового обслуговування, вихідні дані яких мають нести лише рекомендаційний характер.

У процесі розробки даного ПЗ ми покращили навички роботи з мовою програмування C++, розібрались із принципами зручної організації проекту, розміщуючи код кожного з класів в окремих файлах. Це значно спрощує процес відлагодження, сприяє більшій інтуїтивності та наочності, може допомогти іншим розробникам швидше розібратись у чужій для них кодовій базі. Ми широко застосовували принципи ООП, зокрема наслідування та інкапсуляція, розібрались із способами збереження та коректної взаємодії об'єктів між собою. Були вдосконалені навички пошуку та виправлення помилок, які призводили до некоректного завершення програми на певному етапі виконання.

Система масового обслуговування об'єктів виробництва була належним чином протестована шляхом проведення великої кількості експериментів з різними параметрами. Виконавчі файли отримувались за допомогою компіляторів декількох різних виробників, відмічено абсолютну стабільність і надійність реалізованих рішень. На нашу думку, адаптація подібного проекту для імітації процесів відновлення житлових об'єктів не потребує дуже великої кількості часу через наявність вже реалізованих алгоритмів.

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						65
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Будівництво є однією з найважливіших та найпроблемніших галузей людської діяльності. Як і всі інші сфери, воно беззаперечно підлягає впливу інформаційних технологій. Це потребує все більш вдосконалених підходів до вирішення задач не тільки безпосереднього проектування та створення нових об'єктів, а й запобігання виникненню невдалих конструкцій, які псують зовнішній вигляд вулиць, шкодять репутації селищ і окремих людей, здійснюють негативний вплив на екологію. Тому наша дипломна робота присвячена аналізу існуючих рішень у галузі будівництва та розробці нових інструментів для виявлення житлових об'єктів з проблемним станом.

Предметна область цієї галузі є дуже широкою і включає в себе нормативно-законодавчі, технічні, інформаційні аспекти. Відомо, що впродовж останніх років наша держава намагається все більш широко комп'ютеризувати простір навколо об'єктів житлової сфери. До таких спроб належить Єдина державна електронна система у сфері будівництва (ЄДЕСБ). Вона реалізована у вигляді веб-порталу, який надає певні можливості для додавання та пошуку інформації про об'єкти нерухомості. Цілями створення такого реєстру є надання населенню прозорих та достовірних даних про існуючі та майбутні будівельні конструкції, що може сприяти протидії корупції та унеможливленню реалізації певних шахрайських схем. Проаналізувавши роботу цього сервісу, ми стикнулись з деякими проблемами, такими як неможливість знайти дані про деякі існуючі конструкції, відсутність систематизованих відомостей про невдалі об'єкти будівництва у потрібному вигляді.

Зручним ресурсом є портал Київської нерухомості, який надає достовірні дані про об'єкти житлового будівництва та їх стан. Але подібну якісну інформацію нам не вдалось знайти для інших населених пунктів нашої країни.

Відомо, що створення будь-якої системи прогнозування тих чи інших даних потребує велику кількість інформації про існуючі конструкції. Тому для розробки програмного забезпечення з функцією виявлення об'єктів будівництва з проблемним станом ми скористались інформацією з сайту Київської міської ради,

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		66

яка включає в себе детальні дані про незавершені об'єкти. Завантаживши перелік існуючих адрес у нашій столиці із відкритих джерел, ми отримали базу даних, цілком придатну для створення бажаного ПЗ. Проте через відсутність можливості використовувати вміст БД порталу Київської нерухомості, та через неспроможність ручного уведення відомостей про кілька тисяч будинків, ми були вимушені заповнити пусті поля випадковими даними. Достовірною стала лише інформація про невдалі об'єкти, кількість записів якої не перевищує 50.

Після приведення знайдених відомостей до зручного для використання вигляду ми створили необхідний скрипт, який автоматизує заповнення БД під управлінням PostgreSQL. За допомогою засобів веб-програмування, таких як JavaScript і Python, ми створили додаток, який різними способами взаємодіє з існуючою інформацією, а також виконує прогнози шляхом залучення інструментів для аналізу даних. Для розгортання цього проекту була написана детальна інструкція, перелік дій якої включає в себе встановлення деякого ПЗ та робота з командним рядком ОС.

Для тестування створених рішень, зокрема моделей машинного навчання, а також формування загальних висновків про стан будівельної галузі в місті Києві ми створили скрипт на мові програмування Python. Він включає в себе реалізацію різноманітних експериментів над зібраними даними, а також перевіряє реакцію певних інструментів бібліотеки scikit-learn на зміни деяких полів тестової вибірки. Результати випробувань автоматично фіксуються у вигляді графіків, діаграм, csv-файлу разом з даними прогнозів, відомостями про точність роботи використаних моделей. Отримана інформація виглядає цілком закономірною і може повністю відповідати дійсності. Проте ми не радимо використовувати її при прийнятті серйозних рішень по інвестуванню в той чи інший об'єкт будівництва.

Наступним кроком стала розробка ПЗ для моделювання процесів відновлення незавершених об'єктів будівництва. Для цього ми дослідили роботу системи масового обслуговування, яка симулює процеси реального виробництва. Нами були надані та проаналізовані відповідні графічні матеріали, отримані за допомогою табличного процесора. Проте в поточний момент часу подібне

					ЗПІ-зп01.080БАК.005 ПЗ	Арк.
						67
Зм.	Арк.	№ докум.	Підпис	Дата		

рішення для роботи в будівельній галузі знаходиться на етапі розробки. Був створений лише основний каркас, який включає в себе сформовану файлову структуру проекту, інтерфейс користувача, код для введення і виведення даних, реалізацію алгоритму імітації за принципом найближчої події. Недоліком таких рішень є наближеність вихідних даних, отриманих шляхом використання випадкових значень параметрів у заданих діапазонах. Такі відомості часто виглядають правдоподібно і коректно співвідносяться з реальним світом, проте їх практична цінність знаходиться на низькому рівні.

Під час створення описаних додатків ми вдосконалили навички роботи зі скриптовими мовами програмування (Python), набули досвід побудови веб-проектів, навчилися практично використовувати інструменти для аналізу даних та машинного навчання. Крім того, ми поглибили знання в області ООП, опанували деякі можливості мови C++ для вирішення задач у сфері моделювання систем, отримали корисну інформацію про будівельну галузь в нашій державі.

Розроблені рішення мають певний потенціал потрапити в арсенал будівельників. Їх слід вдосконалити шляхом залучення реальної достовірної інформації, щоб покращити результати роботи досліджених нами інструментів роботи з даними. Також потрібно розширити та оптимізувати функціональні можливості веб-додатку через використання більш сучасних підходів до дизайну та серверної частини ПЗ, усунути проблеми із швидкістю, вирішити питання розміщення проекту в мережі Інтернет. Такі кроки можуть в певній мірі сприяти покращенню ситуації в будівельній галузі нашої держави.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Будівництво // Термінологічний словник-довідник з будівництва та архітектури / Р. А. Шмиг, В. М. Боярчук, І. М. Добрянський, В. М. Барабаш; за заг. ред. Р. А. Шмига. – Львів, 2010. — С. 41.
2. Квартира в недострое – как избежать мошенничества.
URL: <https://bucha.com.ua/index.php?newsid=1151069380>
3. Міністерство розвитку громад, територій та інфраструктури України.
URL: <https://mtu.gov.ua/>
4. Основи JavaScript. URL: <https://developer.mozilla.org/>
5. Отримання дозвільних документів на будівництво.
URL: https://wiki.legalaid.gov.ua/index.php/Отримання_дозвільних_документів_на_будівництво#.
6. Офіційний портал Києва. URL: <https://kyivcity.gov.ua/>
7. Офіційний сайт Matplotlib. URL: <https://matplotlib.org/>
8. Офіційний сайт NumPy. URL: <https://numpy.org/>
9. Офіційний сайт pandas. URL: <http://pandas.pydata.org/>
10. Офіційний сайт Scikit-learn. URL: <https://scikit-learn.org>
11. Офіційний сайт seaborn. URL: <https://seaborn.pydata.org/>
12. Офіційний сайт Сайт Django. URL: <https://www.djangoproject.com/>
13. Оформлення права власності на об'єкт незавершеного будівництва.
URL: <https://dozvil.ua/uk/registraciya-nezavershennogo-stroitelstva>
14. Питання прийняття в експлуатацію закінчених будівництвом об'єктів.
URL: <https://zakon.rada.gov.ua/go/461-2011-п>
15. Портал державної електронної системи у сфері будівництва.
URL: <https://e-construction.gov.ua/>
16. Портал Київської нерухомості.
URL: <https://my-realty.kiev.ua/uk/houses/kyev-1/>
17. Порядок розроблення проектної документації на будівництво об'єктів.
URL: <https://web.archive.org/web/20211123034249/>

18. Придбання об'єкта незавершеного будівництва.

URL: https://wiki.legalaid.gov.ua/index.php/Придбання_об'єкта_незавершеного_будівництва#.

19. Ситник В.Ф., Орленко Н.С. Імітаційне моделювання: Навчальний посібник. – К: КНЕУ - 1998. – 230с.

20. Сучасний підручник з JavaScript. URL: <https://uk.javascript.info/>

21. Теорія ймовірностей та математична статистика [Електронний ресурс]: підручник / Т.А. Ліхоузова. – Київ : НТУУ «КПІ ім. Ігоря Сікорського», 2018, с. 173-210.

22. Український довідник CSS. URL: <https://css.in.ua/css/properties>

23. Український довідник HTML. URL: <https://css.in.ua/html/tags>

24. Чи реально інвесторам в Україні захиститись від будівельних афер?
URL: <https://yur-gazeta.com/publications/practice/neruhomist-ta-budivnictvo/chi-realno-investoram-v-ukrayini-zahistitis-vid-budivelnih-afer.html>

25. Як перевірити надійність забудовника? URL: <https://neruhomo.in.ua/yak-pereviriti-nadijnist-zabudovnika/>

ДОДАТОК А

Програма для аналізу даних про об'єкти житлового будівництва (script.py)

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm

def GetArrayFromLinesOfCSVFormat(FileName):
    lines=[]
    file=open(FileName,'r',encoding='UTF8')
    file.readline()
    for line in file:
        lines.append(line)
    file.close()
    arr=[]
    LineNumber=0
    j=0
    for line in lines:
        i=0
        PrevI=0
        arr.append([])
        for j in range(0,5):
            while(line[i]!=';'):
                i+=1
            arr[LineNumber].append(line[PrevI:i])
            i+=1
```

```

        PrevI=i
arr[LineNumber].append([])
while True:
    while(line[i]!=';' and line[i]!=';'):
        i+=1
    arr[LineNumber][5].append(line[PrevI:i])
    if(line[i]==';'):
        break
    i+=1
    PrevI=i
for j in range(0,4):
    i+=1
    PrevI=i
    while(line[i]!=';' and line[i]!='\n'):
        i+=1
    arr[LineNumber].append(line[PrevI:i])
if(arr[LineNumber][9]=="Невдала будівля"):
    arr[LineNumber].append(0)
else:
    arr[LineNumber].append(1)
LineNumber+=1
# arr.sort(key = lambda x: x[10])
return arr
def AddToDictionary(dict,value):
    if (value not in dict):
        dict[value]=1
    else:
        dict[value]+=1
def LoadFromDictionaryToArray(dict):
    arr=[]

```

```

arr.append([])
arr.append([])
for key, value in dict.items():
    arr[0].append(key)
    arr[1].append(value)
return arr

def DrawHorizontalPlot(data,title,FileName):
    fig, ax = plt.subplots()
    sns.barplot(x=data[1], y=data[0])
    fig.set_size_inches(15,10)
    plt.rcParams['font.size'] = '20'
    plt.title(title,fontsize=20)
    plt.tight_layout()
    plt.savefig(FileName)

def DrawPieChart(data,title,FileName):
    for i in range(0,len(data[0])):
        data[0][i]=data[0][i]+" (" +str(data[1][i])+" )"
    fig, ax = plt.subplots()
    ax.pie(data[1],autopct='% 1.1f%% ',pctdistance=1.4,explode=[0.15] + [0 for
_ in range(len(data[0]) - 1)],textprops={'fontsize': 20})
    plt.legend(bbox_to_anchor = (0.22,1),loc = 'upper right', labels =
data[0],fontsize=20)
    plt.title(title,fontsize=20)
    ax.axis("equal")
    fig.set_size_inches(15,10)
    plt.savefig(FileName)

def GetPredictions(DataForPrediction,x_train, x_test, y_train, y_test):
    lgc = LogisticRegression(solver='lbfgs',random_state=0)
    lgc.fit(x_train, y_train)
    LGCPredictions = lgc.predict(DataForPrediction)

```

```

rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
RFPredictions = rfc.predict(DataForPrediction)
SVM = svm.SVC()
SVM.fit(x_train, y_train)
SVMPredictions = SVM.predict(DataForPrediction)
LgsAccuracy=lgc.score(x_test,y_test)
RfcAccuracy=rfc.score(x_test,y_test)
SvmAccuracy=SVM.score(x_test,y_test)
if LgsAccuracy>RfcAccuracy and LgsAccuracy>SvmAccuracy:
    return LGCPredictions
else:
    if RfcAccuracy>LgsAccuracy and RfcAccuracy>SvmAccuracy:
        return RFPredictions
    return SVMPredictions
def GetBestPredictions(DataForPrediction,x_train, x_test, y_train, y_test):
    predictions=GetPredictions(DataForPrediction,x_train,  x_test,  y_train,
y_test)
    CountOfFinishedConstructions=0
    CountOfUnfinishedConstructions=0
    for i in predictions:
        if i==0:
            CountOfUnfinishedConstructions+=1
        else:
            CountOfFinishedConstructions+=1
    return (CountOfUnfinishedConstructions*100.0)/len(DataForPrediction)
def
BuildPlotOfPredictions(DataForPrediction,x_train,x_test,y_train,y_test,steps,title,Colu
mnName,LabelY,FileName):
    predictions=[]

```

```

AverageChangedValuesForPlot=[]
ChangedDataForPrediction=[]
for j in range(0,len(steps)):
    DataFrameCopy = DataForPrediction.copy()
    ChangedDataForPrediction.append(DataFrameCopy)
    predictions.append([])
for i in range(0,10):
    AverageChangedValues=[]
    for j in range(0,len(steps)):
        AverageChangedValue=0.0
        for k in range(0,len(ChangedDataForPrediction[j])):
            AverageChangedValue+=float(ChangedDataForPrediction[j][ColumnName][k])
        AverageChangedValues.append(AverageChangedValue/len(ChangedDataForPrediction[j]))

    AverageChangedValueForPlot=0.0
    for j in range(0,len(steps)):
        AverageChangedValueForPlot+=AverageChangedValues[j]

    AverageChangedValuesForPlot.append(AverageChangedValueForPlot/len(steps)
)

    for j in range(0,len(steps)):

        predictions[j].append(GetBestPredictions(ChangedDataForPrediction[j],x_train,
x_test, y_train, y_test))

        for j in range(0,len(steps)):
            for k in range(0,len(ChangedDataForPrediction[j])):

                ChangedDataForPrediction[j][ColumnName][k]=str(float(ChangedDataForPrediction[j][ColumnName][k])+float(steps[j]))

        plt.figure(figsize=(15,15))

```



```

        for i in range(0,len(steps)):
            plt.plot(predictions[i],AverageChangedValuesForPlot,label="step
"+str(steps[i]))

plt.title(title,fontsize=30)
plt.xlabel('Процент недобудов',fontsize=30)
plt.ylabel(LabelY,fontsize=30)
plt.rcParams['font.size'] = '30'
plt.tick_params(axis='both', which='major', labelsize=30)
plt.legend()
plt.savefig("Plot "+FileName)
plt.clf()

AverageX=[]
AverageY=[]
LegendTitles=[]
EqualX={ }

for i in range(0,len(steps)):
    AverageValue=0
    for j in range(0,len(predictions[i])):
        AverageValue+=predictions[i][j]

AverageX.append(float(str("{:.2f}".format(float(AverageValue/len(predictions[i]
))))))

    if(AverageX[i] not in EqualX):
        EqualX[int(AverageValue/len(predictions[i]))]=AverageX[i]
        AverageValue=0
        for j in range(0,len(ChangedDataForPrediction[i])):

AverageValue+=float(ChangedDataForPrediction[i][ColumnName][j])

```

```

AverageY.append(float(str("{:.2f}".format(float(AverageValue/len(ChangedData
ForPrediction[i]))))))

LegendTitles.append("step "+str(steps[i]))
while len(AverageY)>len(EqualX):
    AverageX.pop(0)
    AverageY.pop(0)
    LegendTitles.pop(0)
fig, ax = plt.subplots()
sns.barplot(x=AverageX,y=AverageY,label=LegendTitles)
fig.set_size_inches(15,15)
plt.rcParams['font.size'] = '30'
plt.title(title,fontsize=30)
plt.tick_params(axis='both', which='major', labelsize=30)
plt.legend()
ax.set_xlabel('Процент недобудов',fontsize=30)
ax.set_ylabel(LabelY,fontsize=30)
plt.savefig("Barplot "+FileName)
plt.clf()

DataForLearning=GetArrayFromLinesOfCSVFormat("DataForLearning.csv")
DataForPrediction=GetArrayFromLinesOfCSVFormat("DataForPrediction.csv")
DataFrameForLearning=pd.DataFrame(DataForLearning,columns=['Вулиця','Кі
лькість недобудов на вулиці','№ буд.','Рік','Кількість недобудов за
рік','Матеріали','Рейтинг матеріалів','Кількість поверхів','Висота стелі','Стан','Стан
(bool)'])

DataFrameForPrediction=pd.DataFrame(DataForPrediction,columns=['Вулиця','
Кількість недобудов на вулиці','№ буд.','Рік','Кількість недобудов за
рік','Матеріали','Рейтинг матеріалів','Кількість поверхів','Висота стелі','Стан','Стан
(bool)'])

y = DataFrameForLearning.iloc[:,10].values

```

```

DataFrameForLearning.drop(DataFrameForLearning.columns[[0,2,3,5,9,10]],axis=1,inplace=True)

x = DataFrameForLearning.iloc[:,0:]

DataFrameForPrediction.drop(DataFrameForPrediction.columns[[0,2,3,5,9,10]],axis=1,inplace=True)

x_train,x_test,y_train,y_test,=train_test_split(x,y,random_state=0)

BuildPlotOfPredictions(DataFrameForPrediction,x_train,x_test,y_train,y_test,[1,2,3],"Залежність процентної частки недобудов\пвід кількості незавершених об'єктів на вулиці","Кількість недобудов на вулиці","Середня кількість недобудов на вулиці","StreetRating")

BuildPlotOfPredictions(DataFrameForPrediction,x_train,x_test,y_train,y_test,[1,2,3],"Залежність процентної частки недобудов\пвід кількості незавершених об'єктів за рік","Кількість недобудов за рік","Середня кількість недобудов за рік","YearRating")

BuildPlotOfPredictions(DataFrameForPrediction,x_train,x_test,y_train,y_test,[1,2,3],"Залежність процентної частки недобудов\пвід рейтингу матеріалів","Рейтинг матеріалів","Середній рейтинг матеріалів","MaterialRating")

BuildPlotOfPredictions(DataFrameForPrediction,x_train,x_test,y_train,y_test,[1,2,3],"Залежність процентної частки недобудов\пвід кількості поверхів","Кількість поверхів","Середня кількість поверхів","CountOfStoreys")

BuildPlotOfPredictions(DataFrameForPrediction,x_train,x_test,y_train,y_test,[1,2,3],"Залежність процентної частки недобудов\пвід висоти стелі","Висота стелі","Середня висота стелі","CeilingHeight")

StreetsDict={ }

YearsDict={ }

MaterialsDict={ }

NumbersOfStoreysDict={ }

HeightsOfCeilingsDict={ }

StatesDict={ }

for i in range(0,len(DataForLearning)):

```

```

if(DataForLearning[i][10]==0):
    AddToDictionary(StreetsDict,DataForLearning[i][0])
    AddToDictionary(YearsDict,DataForLearning[i][3])
    for j in DataForLearning[i][5]:
        AddToDictionary(MaterialsDict,j)
    AddToDictionary(NumbersOfStoreysDict,DataForLearning[i][7])
    AddToDictionary(HeightsOfCeilingsDict,DataForLearning[i][8])
    AddToDictionary(StatesDict,DataForLearning[i][9])
for i in range(0,len(DataForPrediction)):
    AddToDictionary(StatesDict,DataForPrediction[i][9])
streets=LoadFromDictionaryToArray(StreetsDict)
years=LoadFromDictionaryToArray(YearsDict)
materials=LoadFromDictionaryToArray(MaterialsDict)
NumbersOfStoreys=LoadFromDictionaryToArray(NumbersOfStoreysDict)
HeightsOfCeilings=LoadFromDictionaryToArray(HeightsOfCeilingsDict)
states=LoadFromDictionaryToArray(StatesDict)
LogisticRegression = LogisticRegression(solver='lbfgs',random_state=0)
LogisticRegression.fit(x_train, y_train)
LogisticRegressionPredictions =
LogisticRegression.predict(DataFrameForPrediction)
RandomForest = RandomForestClassifier()
RandomForest.fit(x_train,y_train)
RandomForestPredictions = RandomForest.predict(DataFrameForPrediction)
SupportVectorMachine = svm.SVC()
SupportVectorMachine.fit(x_train, y_train)
SupportVectorMachinePredictions =
SupportVectorMachine.predict(DataFrameForPrediction)
LGStatesDict={ }
RFStatesDict={ }
SVMStatesDict={ }

```

```

FileOfPredictions = open("predictions.csv", "w", encoding='utf-8-sig')
FileOfPredictions.write("Вулиця;Кількість недобудов на вулиці;№
буд.;Рік;Кількість недобудов за рік;Матеріали;Рейтинг матеріалів;Кількість
поверхів;Висота стелі;Стан;Логістична регресія;Випадковий ліс;Метод опорних
векторів\n")
for i in range(0,len(DataForPrediction)):
    FileLine=""
    for j in range(0,len(DataForPrediction[i])):
        FileLine=FileLine+str(DataForPrediction[i][j])+";"
    if(LogisticRegressionPredictions[i]==1):
        AddToDictionary(LGStatesDict,"Успіх")
        FileLine=FileLine[:len(FileLine)-2]+"Успіх;"
    else:
        AddToDictionary(LGStatesDict,"Недобудова")
        FileLine=FileLine[:len(FileLine)-2]+"Недобудова;"
    if(RandomForestPredictions[i]==1):
        AddToDictionary(RFStatesDict,"Успіх")
        FileLine=FileLine+"Успіх;"
    else:
        AddToDictionary(RFStatesDict,"Недобудова")
        FileLine=FileLine+"Недобудова;"
    if(SupportVectorMachinePredictions[i]==1):
        AddToDictionary(SVMStatesDict,"Успіх")
        FileLine=FileLine+"Успіх\n"
    else:
        AddToDictionary(SVMStatesDict,"Недобудова")
        FileLine=FileLine+"Недобудова\n"
    FileOfPredictions.write(FileLine)
FileOfPredictions.close()
LGStates=LoadFromDictionaryToArray(LGStatesDict)

```

```

RFStates=LoadFromDictionaryToArray(RFStatesDict)
SVMStates=LoadFromDictionaryToArray(SVMStatesDict)
DrawHorizontalPlot(streets,"Статистика недобудов за вулицями","Statistic
streets")
DrawHorizontalPlot(years,"Статистика недобудов за роками","Statistic years")
DrawHorizontalPlot(materials,"Статистика недобудов за використаними
матеріалами","Statistic materials")
DrawHorizontalPlot(NumbersOfStoreys,"Статистика недобудов за кількістю
поверхів","Statistic NumbersOfStoreys")
DrawHorizontalPlot(HeightsOfCeilings,"Статистика недобудов за висотою
стелі","Statistic HeightsOfCeilings")
DrawHorizontalPlot(states,"Статистика недобудов за станом","Statistic
states")
DrawPieChart(states,"Співвідношення зданих в експлуатацію будівель та
недобудов","Statistic states")
DrawPieChart(LGStates,"Спрогнозоване співвідношення зданих в
експлуатацію будівель та недобудов\n(логістична регресія)","Statistic
logisticRegression")
DrawPieChart(RFStates,"Спрогнозоване співвідношення зданих в
експлуатацію будівель та недобудов\n(випадковий ліс)","Statistic RandomForest")
DrawPieChart(SVMStates,"Спрогнозоване співвідношення зданих в
експлуатацію будівель та недобудов\n(метод опорних векторів)","Statistic
SupportVectorMachine")
print("The file predictions.csv was created")
print("Logistic                regression                accuracy:
"+str(LogisticRegression.score(x_test,y_test)))
print("Random forest accuracy: "+str(RandomForest.score(x_test,y_test)))
print("Support                vector                machine                accuracy:
"+str(SupportVectorMachine.score(x_test,y_test)))

```


					ЗП-зп01.080БАК.005 ДЗ				
					Тема. Схема бази даних об'єктів житлового будівництва	Літ.		Маса	Масштаб
		№ докум.	Підпис						
Розробив	Кононов М.								
Перевірив	Лісовиченко О.								
					Арк.	1	Аркушів 5		
						КПІ ім. Ігоря Сікорського Група ЗП-зп01			
Затв.									

					ЗПІ-зп01.080БАК.005 Д5				
					Тема. Алгоритми роботи ПЗ для моделювання процесів відновлення невдалих об’єктів будівництва	Літ.		Маса	Масштаб
					Арк. 1		Аркушів 5		
					КПІ ім. Ігоря Сікорського Група ЗПІ-зп01				
Затв.									

