Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Факультет інформаційних технологій

(факультет)

Кафедра системного аналізу та управління

(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня магістра

Студента П’ятоволенка Олександра Олександровича

академічної групи 124м – 22 – 1

спеціальності 124 Системний аналіз

на тему: «Аналіз показників гри жанру пошук предметів та оптимізація ігрових характеристик для підвищення платежів»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**Інституційною**	
кваліфікаційної роботи	*к.ф.-м.н., доц. Хом’як Т.В.*			
розділів:				
Інформаційно-аналітичний	*к.ф.-м.н., доц. Хом’як Т.В.*			
Спеціальний розділ	*к.ф.-м.н., доц. Хом’як Т.В.*			
Експериментально-аналітичний	*к.ф.-м.н., доц. Хом’як Т.В.*			
Рецензент	*д.т.н.,проф. Алексєєв М.А.*			
Нормоконтролер	*к.ф.-м.н., доц. Хом’як Т.В.*			

Дніпро

2023

ЗАТВЕРДЖЕНО
завідувач кафедри

Системного аналізу та управління

(повна назва)

_____ *к.т.н., доц. Желдак Т.А.*

(підпис) (прізвище, ініціали)

«____» _____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *магістра*
студенту П'ятоволенку О.О. **академічної групи** *124м – 22 1*

спеціальності: *124 Системний аналіз*

на тему «Аналіз показників гри жанру пошук предметів та оптимізація ігрових характеристик для підвищення платежів»
затверджену наказом ректора НТУ «Дніпровська політехніка»
від _____

Розділ	Зміст	Терміни виконання
1. Інформаційно-аналітичний розділ	*Проаналізувати структуру об'єкта дослі- дження. Визначити предметну область дослідження та проблему, що розв'язується. Обґрунтувати методи виконання поставлених завдань.*	
2.Спеціальний розділ	*Розв'язати поставлені задачі: визначити відмінності двох класів гравців та на основі цього визначити нові ігрові налаштування.*	
3.Експериментально-аналітичний розділ	*Визначити нульову та альтернативну гіпотези, провести A/B–тестування, заміряти результати та зробити висновки щодо доцільності впровадження нових налаштувань.*	

Завдання видано _____ *к.ф.-м.н., доц. Хом'як Т.В.*

Дата видачі: _____

Дата подання до екзаменаційної комісії: _____

Прийнято до виконання _____ П'ятоволенко О.О.

(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 88 с., 49 рис., 17 табл., 4 додатки, 25 джерел.

Об'єктом дослідження в роботі є гра жанру пошук предметів з ціллю визначення нових налаштувань для підвищення платежів.

Предметом дослідження є показники гри жанру пошук предметів та показники гравців.

Метою даної кваліфікаційної роботи є визначення закономірностей між двома класами платячих та неплатячих гравців та підбір стратегії оптимізації геймплейних механік для підвищення рівня платежів гравців.

Методи дослідження: Random Forest, Support Vector Machines, Logistic Regression – алгоритми для спроби вирішити задачу класифікації, K-means – для кластерного аналізу, A/B-тестування – для визначення чи вплинули позитивно нові налаштування відносно старих.

В *інформаційно–аналітичному розділі* наведено аналіз об'єкту дослідження та ключових проблем на ньому. Поставлені задачі дослідження та обрано концепції їх розв'язання.

У *спеціальному розділі* визначені основні відмінності між класами гравців, визначені нові ігрові налаштування. В результаті визначення налаштувань проведено A/Bтестування та зроблені висновки, щодо доцільності впровадження налаштувань для всіх гравців.

Практична цінність отриманих результатів полягає в тому, що результати цього дослідження можуть послужити важливим джерелом інформації для розробників ігор, маркетологів та дослідників, що працюють у сфері відеоігор, а також для академічної спільноти, яка вивчає вплив ігор на суспільство та культуру.

Ключові слова: ПОШУК ПРЕДМЕТІВ, ГРА, КЛАСИФІКАЦІЯ, КЛАСТЕРИЗАЦІЯ, МАШИННЕ НАВЧАННЯ, A/B-ТЕСТУВАННЯ.

ABSTRACT

Explanatory note: 88 p., 49 figs., 17 tabl., 4 add., 25 sources.

The object of research in the work is a game of the hidden objects genre with the aim of determining new settings for increasing payments.

The subject of the study is the indicators of the hidden objects game and the indicators of the players.

The purpose of this diploma project is to determine the optimal strategy for optimizing gameplay mechanics to increase payments.

Research methods: The Random Forest, Support Vector Machines and Logistic Regression algorithms are used for trying to solve the classification problem, The K-means algorithm is used for cluster analysis, The A/B-testing method is used for determining whether the new settings had a positive effect relative to the old ones.

The informational and analytical section provides an analysis of the research object and its key problems. Research tasks are set and concepts for their solution are chosen.

In a special section, the main differences between player classes are defined, and new game settings are defined. As a result of determining the settings, A/B testing was conducted and conclusions were drawn regarding the feasibility of implementing the settings for all players.

The practical value of the obtained results is that the results of this study can serve as an important source of information for game developers, marketers and researchers working in the field of video games, as well as for the academic community studying the impact of games on society and culture.

Keywords: HIDDEN OBJECTS, GAME, CLASSIFICATION, CLASTERISTION, MACHINE LEARNING, A/B-TESTING.

ЗМІСТ

ВСТУП

У сучасному світі ігрова індустрія виявляє неабиякий вплив на економіку та культуру суспільства. З кожним роком вона розширюється та вдосконалюється, завойовуючи нові сегменти ринку та привертаючи мільйони гравців із різних куточків світу. Ігри жанру «Пошук Предметів» (Hidden Object Games) визначаються як один із найпопулярніших жанрів у галузі відеоігор. Цей жанр привертає гравців своєю захоплюючою геймплейною механікою, загадковим сюжетом та можливістю розвивати інтелектуальні навички.

Зростання популярності цього жанру призвело до появи нових викликів для розробників ігор. Оптимізація геймплейних характеристик та аналіз показників стали важливою задачею для студій, які прагнуть не лише задовольнити потреби гравців, а й максимізувати свій прибуток через оптимізацію монетизації гри. Справжня розгадка полягає в збалансуванні між цікавістю гравців та фінансовою стабільністю розробників, яка може бути досягнута за допомогою детального аналізу і оптимізації ігрових характеристик.

Ця наукова робота присвячена вивченню різних аспектів жанру «Пошук Предметів» з метою визначення оптимальних стратегій оптимізації геймплейних характеристик для підвищення рівня платежів гравців. Також на меті стоїть спроба вирішити задачу класифікації для даного набору даних та провести кластерний аналіз. Шляхом аналізу попиту гравців, їхньої взаємодії з ігровим середовищем та реакції на різні геймплейні елементи, стоїть мета розкрити закономірності, які впливають на їхню вартість для гравців та рентабельність для розробників.

Метою даної кваліфікаційної роботи є визначення закономірностей між двома класами платячих та неплатячих гравців та підбір стратегії оптимізації геймплейних механік для підвищення рівня платежів гравців.

Об’єктом дослідження в роботі є гра жанру пошук предметів з ціллю визначення нових налаштувань для підвищення платежів.

Предметом дослідження є показники гри жанру пошук предметів та показники гравців.

Методи дослідження*:* Random Forest, Support Vector Machines, Logistic Regression – алгоритми для спроби вирішити задачу класифікації, K-means – для кластерного аналізу, A/B-тестування – для визначення чи вплинули позитивно нові налаштування відносно старих, z-тест пропорцій – статистичний тест для порівняння пропорцій із двох вибірок, мова Python та допоміжні її бібліотеки – для рутинних задач.

Теоретичне значення кваліфікаційної роботи полягає в тому, що отримані результати можуть бути використані розробниками ігор подібного жанру.

Нові наукові рішення , які були запропоновані особисто дослідником:

- Розбиття неплатячого класу на кластери та встановлення динамічної складності для кожного з них. Інформації про потенційні кластери гравців та їх імплементацію гри жанру HO в доступній мережі знайдені не було, тому можна вважати, що результати одержано вперше.
- Оформлення кластерної моделі у вигляді API для визначення кластера у конкретного гравця. Таких прикладів багато, тому можна вважати, що це рішення було вдало адаптовано.

Практичне значення отриманих результатів полягає в застосуванні підходу визначення кластера гравця та визначення динамічної складності гри на практиці у вигляді проведення A/B – тестування. Після проведення тестування був проведений статистичний тест, який показує доцільність використання нового підходу на практиці для всіх гравців.

Робота брала участь у II турі Всеукраїнського конкурсу студентських наукових робіт зі штучного інтелекту 2023 в напрямі «Аналіз даних». Також робота була оприлюднена на XVII Міжнародній науково-практичній конференції «МОДЕЛЮВАННЯ ТА ПРОГНОЗУВАННЯ ЕКОНОМІЧНИХ ПРОЦЕСІВ».

Ця робота відзначається важливістю своєї теми в контексті розвитку ігрової індустрії та її впливу на економіку. Отже, результати цього дослідження можуть послужити важливим джерелом інформації для розробників ігор, маркетологів та дослідників, що працюють у сфері відеоігор, а також для академічної спільноти, яка вивчає вплив ігор на суспільство та культуру.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Бустер – ігровий бонус, який облегшує проходження рівня;

Hint – бустер підказка, який підсвічує один випадковий предмет;

Bomb – бустер бомба, який підриває три предмети з рівня;

Freeze – бустер заморожування, який заморожує час на рівня;

Torch – бустер ліхтар, який підсвічує затемнені предмети;

HO – Hidden Objects. Гра жанру пошук предметів;

Геймплей – ігровий процес. Термін, яким називають особливості взаємодії людини з відеогрою. Ці особливості створюються за допомогою правил, завдань та способів їх розв'язування, які пропонує гра.

РОЗДІЛ 1
ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ

1.1 Загальні відомості

Опис гри

Сенс гри полягає в тому, що є пейзаж на якому розташовані предмети. Приклад такого рівня зображений на рис. 1. Є панель зліва, на якій зображені предмети, які потрібно знайти гравцю. Рівень обмежений часом та має допоміжні бонуси, такі як:

1. Hint – підсвічує один з предметів на полі;
2. Bomb – підриває три предмети з поля, які більше не потрібно знаходити;
3. Freeze – заморожує час;
4. Torch – бувають ігри, коли замість предметів на лівій панелі зображена тільки їх тінь. Бонус Torch відсвітлює ці предмети.
5. Extra time – надається змога докупити час, якщо час був вичерпаний, а гравець не встиг знайти всі предмети.

Рис. 1 – Вікно рівня гри Hidden Objects

Також присутня система покарань за закликання ігрового поля. Якщо гравець починає швидко клацати поле з метою випадково зібрати предмет, то спочатку знімається невеличка кількість монет з балансу. Якщо це продовжується далі, то з'являється вікно, яке інформує гравця про те, що треба сплатити штраф, щоб продовжити грати, або рівень буде завершено.

Крім бонусів, також є ігрові ресурси, такі як монети та енергія. Розпочати грати рівень коштує 1 енергію. Енергія, також може відновитися за 10 хвилин, але максимальна кількість енергій, яка може відновитися дорівнює 5. Монети можна витрачати для того, щоб купувати бонуси та купляти енергію. Монети можна заробляти за різні активності, такі як: пройдений рівень, зібране досягнення, посідання високого місця в рейтингу, отримання різних призів за прогрес та інші. Всі ці активності будуть більш детально розглянуті при аналізі.

Також присутній магазин, в якому гравець може придбати додаткові ігрові ресурси за гроші.

1.2 Вихідні налаштування гри

Назва	**Ціна**	**Додаткова інформація**
Extra time	500	Дається додаткові 15 сек.
Hint	750	-
Bomb	1000	-
Freeze	2000	-
Torch	1400	-
Penalty	500	Покарання, за багато хибних кліків.
Time	75.5	Середній час, який дається на проходження рівня. **(*)**
Energy	1000	Ресурс, який знімається за початок рівня.

Level Items	17.8	17.8 – в середньому предметів на рівні
Level complete	160	Нагорода, яка дається за проходження рівня.
Star chest	645	Сундук, який дається за збір 15 зірок.
Rewarded ad	150	Бонус, який дається за перегляд реклами. Всього можна подивитися рекламу 5 разів на день.
Day bonus	790	Бонус, який дається за вхід в гру.
Rating Prize	6000	Середня кількість монет, яку можна отримати в змаганнях рейтингу.
Mini task reward	35	Нагорода за виконання невеличких завдань.
Time bonus	50	Раз на 4 години можна зібрати цей бонус.
Chest buy	750	Сундук, який дається за проходження рівня. Щоб його відкрити, треба почекати 4 години, або заплатити 750 монет.
Dark level	0.125	0.125 – частота рівнів dark. **(**)**

(*) – Час рівня призначається по черзі та по кругу за цим масивом [80,90,70,60,65,80,75,90,85,60].

(**) – dark рівні – це рівні, коли на лівій панелі замість предметів їх тінь.

1.3 Гіпотези, які треба перевірити під час аналізу

Були зроблені наступні припущення, щодо гравців, які використовують покупки та які треба перевірити під час аналізу:

1. **Час першої ігрової сесії.**

   Було зроблено припущення, що гравці, які проходять більше рівнів за найпершу ігрову сесію, мають більше шансів зробити покупку в грі.

2. **Win-rate.**

   Win-rate - це доля виграшів до всіх стартів рівнів. Було зроблено припущення, що гравці, які робили хоча б одну покупку, мають менший win-rate перед своєю першою покупкою, ніж ті, хто не платить.

3. **Кількість пройдених рівнів в день.**

   Було зроблено припущення, що гравці які платять, мають більше пройдених рівнів перед своєю першою покупкою, ніж ті, хто не платить.

4. **Середня кількість монет, які гравець витрачає або заробляє за один пройдений рівень.**

   Було зроблено припущення, що гравці які платять, витрачають ігрових ресурсів більше ніж заробляють перед своєю першою покупкою, ніж ті, хто не платить.

5. **Кількість бонусів, які гравець витрачає на рівні.**

   Було зроблено припущення, що гравці які платять, витрачають більше бонусів перед своєю першою покупкою, ніж ті, хто не платить. Це відбувається через те, що гра для таких гравців більш складна.

1.4 Побудова моделей машинного навчання

Одною із цілей цієї роботи є спроба вирішити задачу класифікації та побудувати математичну модель, яка зможе розпізнавати гравця, який платить або не платить, на основі його шаблону гри.

Для побудови моделі будуть розглянуті алгоритми Random Forest, Logistic Regression та Support Vector Machines.

Пріоритет одразу ж віддається алгоритму машинного навчання Випадковий Ліс або Random Forest. Вибір впав саме на цей алгоритм, тому що Random Forest історично показує гарні результати при навчанні та простий в розумінні. Інші алгоритми були взяті для порівняння метрик продуктивності.

Для визначення найбільш оптимальних гіперпараметрів буде застосований метод GridSearchCV. Це техніка для пошуку найкращих значень параметрів із заданого набору сітки параметрів. По суті, це метод перехресної перевірки. потрібно ввести модель і параметри. Витягуються найкращі значення параметрів, а потім робляться прогнози.

Для визначення мультиколінеарності у незалежних змінних буде використаний VIF (Variance inflation factor). Якщо у якоїсь із змінних VIF буде більше 10, тоді це означає, що присутня мультиколінеарність.

Для розбиття вибірки на тестову та тренувальну буде використаний метод train_test_split модуля sklearn. На тестову вибірку буде виділено 30% загальної вибірки. Так як класи незбалансовані, буде задіяний метод розставлення вагів.

Для визначення ефективності математичної моделі буде наданий пріоритет метриці AUC score. Він набуває значень від 0.5 до 1 (все включно), де 0.5 означає, що модель нічим не краща випадкової моделі, а 1 означає, що це ідеальна модель. За офіційною документацією цієї метрики визначено, що мінімальний поріг на тестових даних для того, щоб модель вважалась релевантною є 0.7. Модель з AUC від 0.7 до 0.8 вважається прийнятною, від 0.8 до 0.9 вважається гарною, а від 0.9 до 1 відмінною.

Для кластерного аналізу буде використаний алгоритм K-means. Цей алгоритм був обраний тому що він простий в реалізації, він легко адаптується до нових прикладів та може узагальнюватися на кластери різних форм, наприклад на еліптичні кластери.

1.5 Використані технології

Аналіз та побудова моделей виконувалися в середовищі Jupyter Notebook за допомогою мови Python та допоміжних бібліотек. Окрім мови Python були використані наступні бібліотеки та пакети:

1. numpy;
2. pandas;
3. matplotlib;
4. seaborn;
5. joblib;
6. statsmodels.stats.outliers_influence.variance_inflation_factor;
7. scipy.stats;
8. statsmodels.api;
9. phik;
10. mpl_toolkits.mplot3d.Axes3D;
11. sklearn.model_selection.train_test_split;
12. sklearn.preprocessing.MinMaxScaler;
13. sklearn.model_selection.GridSearchCV;
14. sklearn.ensemble.RandomForestClassifier;
15. sklearn.metrics.confusion_matrix;
16. sklearn.metrics.classification_report;
17. sklearn.metrics.ConfusionMatrixDisplay;
18. sklearn.metrics.accuracy_score;
19. sklearn.metrics.roc_curve;
20. sklearn.metrics.RocCurveDisplay;
21. sklearn.metrics.auc;

РОЗДІЛ 2

СПЕЦІАЛЬНИЙ

2.1 Розвідувальний аналіз даних

2.1.1 Інтерпретація стовпців датасету

```
RangeIndex: 1481 entries, 0 to 1480
Data columns (total 62 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              1481 non-null   int64
 1   achievement_reward_collect      1481 non-null   float64
 2   achievement_reward_collect_total 1481 non-null  float64
 3   avg_day_levels_passed           1480 non-null   float64
 4   avg_day_levels_passed_total     1480 non-null   float64
 5   avg_level_time                  1480 non-null   float64
 6   avg_win_rate                    1481 non-null   float64
 7   avg_win_rate_total              1481 non-null   float64
 8   bomb_used                       1243 non-null   float64
 9   bomb_used_total                 1243 non-null   float64
 10  booster_buy                     1462 non-null   float64
 11  booster_buy_total               1462 non-null   float64
 12  browser_name                    1481 non-null   object
 13  buy_energy                      642 non-null    float64
 14  buy_energy_total                642 non-null    float64
 15  chest_open_collect              1050 non-null   float64
 16  chest_open_collect_total        1050 non-null   float64
 17  current_bomb                    1481 non-null   float64
 18  current_energy                  1481 non-null   float64
 19  current_freeze                  1481 non-null   float64
 20  current_hint                    1481 non-null   float64
 21  current_money                   1481 non-null   float64
 22  current_torch                   1481 non-null   float64
 23  day_bonus_collect               1282 non-null   float64
 24  day_bonus_collect_total         1282 non-null   float64
 25  day_levels_passed_new           1481 non-null   float64
 26  day_levels_passed_total         1481 non-null   float64
 27  extra_time_used                 799 non-null    float64
 28  extra_time_used_total           799 non-null    float64
 29  first_payment_level             1481 non-null   float64
 30  freeze_used                     971 non-null    float64
 31  freeze_used_total               971 non-null    float64
 32  gender                          1481 non-null   object
 33  hint_used                       1392 non-null   float64
 34  hint_used_total                 1392 non-null   float64
 35  level_complete_collect          1479 non-null   float64
 36  level_complete_collect_total    1479 non-null   float64
 37  level_failed_missclick          913 non-null    float64
 38  level_failed_quit               61 non-null     float64
 39  level_failed_time               1399 non-null   float64
 40  level_failed_total              1481 non-null   float64
 41  lost                            1481 non-null   int64
 42  mini_tasks_reward_collect       1377 non-null   float64
 43  mini_tasks_reward_collect_total 1377 non-null   float64
 44  new_user_entry_point            1481 non-null   object
 45  pay_penalty                     1449 non-null   float64
 46  pay_penalty_total               1449 non-null   float64
 47  rating_prize_collect            1391 non-null   float64
 48  rating_prize_collect_total      1391 non-null   float64
 49  star_chest_collect              1274 non-null   float64
 50  star_chest_collect_total        1274 non-null   float64
 51  start                           1481 non-null   int64
 52  time_bonus_collect              1428 non-null   float64
 53  time_bonus_collect_total        1428 non-null   float64
 54  torch_used                      576 non-null    float64
 55  torch_used_total                576 non-null    float64
 56  total_levels_24h                1065 non-null   float64
 57  total_payments                  1481 non-null   float64
 58  video_reward_collect            258 non-null    float64
 59  video_reward_collect_total      258 non-null    float64
 60  win                             1481 non-null   int64
 61  is_paying                       1481 non-null   int64
dtypes: float64(54), int64(5), object(3)
memory usage: 717.5+ KB
```

Рис. 2 – загальна інформація про стовпці датасету

Всього в начальному датасеті 62 стовпці, 1481 строки. На рис. 2 зображена загальна інформація. Всі типи даних з самого початку розставлені правильно. В деяких місцях є пропуски.

Є стовпці, які в назві відрізняються тільки суфіксом «_collect». Відмінність в тому, що в стопці без суфіксу для платячих гравців враховані здобутки до його першої покупки, а в стопці з суфіксом враховані загальні здобутки. Для гравців, які ніколи не платили інформація в обох стовпцях однакова. Для навчання будуть використана тільки інформація до моменту першої покупки, загальна інформація тільки для виявлення можливих закономірностей.

Кожен стовпець означає наступне:

1. is_paying – таргет змінна, яка визначає чи платив колись гравець. 1 – так, 0 – ні;
2. id – ідентифікатор;
3. achievement_reward_collect – скільки зароблено монет за досягнення до першої покупки;
4. achievement_reward_collect_total – скільки зароблено монет за досягнення загалом;
5. avg_day_levels_passed – скільки в середньому проходить гравець до першої покупки;
6. avg_day_levels_passed_total - скільки в середньому проходить гравець загалом;
7. avg_level_time – скільки гравець в середньому витрачає часу на проходження рівня;
8. avg_win_rate – win-rate гравця до першої покупки;
9. avg_win_rate_total – win-rate гравця загалом;
10. bomb_used – скільки задіяно бомб до першої покупки;
11. bomb_used_total - скільки задіяно бомб загалом;
12. booster_buy – скільки витрачено на бонуси до першої покупки;
13. booster_buy_total - скільки витрачено на бонуси загалом;

14. browser_name – ім’я браузеру;
15. buy_energy – скільки витрачено на енергію до першої покупки;
16. buy_energy_total – скільки витрачено на енергію загалом;
17. chest_open_collect – скільки накопичено монет з сундука до першої покупки;
18. chest_open_collect_total – скільки накопичено монет з сундука загалом;
19. current_bomb – скільки наразі у гравця бомб;
20. current_hint – скільки наразі у гравця хінтів;
21. current_freeze – скільки наразі у гравця заморозок;
22. current_torch – скільки наразі у гравця факелів;
23. current_money - скільки наразі у гравця монет;
24. day_bonus_collect – скільки накопичено монет за щоденний бонус до першої покупки;
25. day_bonus_collect_total – скільки накопичено монет за щоденний бонус загалом;
26. day_levels_passed_new – скільки пройдено рівнів до першої покупки (був перейменований в day_levels_passed);
27. day_levels_passed_total – скільки пройдено рівнів загалом;
28. extra_time_used – скільки задіяно додаткового часу до першої покупки;
29. extra_time_used_total - скільки задіяно додаткового часу загалом;
30. first_payment_level – перший рівень на якому здійснена покупка;
31. freeze_used – скільки задіяно заморозок до першої покупки;
32. freeze_used_total – скільки задіяно заморозок загалом;
33. gender – гендер;
34. hint_used – скільки задіяно хінтів до першої покупки;
35. hint_used_total – скільки задіяно хінтів загалом;
36. level_complete_collect – скільки зароблено монет за проходження рівнів до першої покупки;

37. level_complete_collect_total – скільки зароблено монет за проходження рівнів загалом;
38. level_failed_missclick – скільки програно рівнів через місклік;
39. level_failed_quit – скільки програно рівнів через вихід;
40. level_failed_time – скільки програно рівнів через час;
41. level_failed_total – скільки програно рівнів загалом;
42. lost - скільки програно рівнів загалом;
43. mini_tasks_reward_collect – скільки зароблено монет за виконання міні-завдань до першої покупки;
44. mini_tasks_reward_collect_total – скільки зароблено монет за виконання міні-завдань загалом;
45. new_user_entry_point – точка входу у найперший момент.
46. pay_penalty – скільки заплачено штрафів за місклік до першої покупки;
47. pay_penalty_total - скільки заплачено штрафів за місклік загалом;
48. rating_prize_collect – скільки зароблено монет за рейтинг до першої покупки;
49. rating_prize_collect_total – скільки зароблено монет за рейтинг загалом;
50. star_chest_collect – скільки зароблено монет за зірковий сундук до першої покупки;
51. star_chest_collect_total – скільки зароблено монет за зірковий сундук загалом;
52. start – скільки всього було стартів рівнів;
53. time_bonus_collect – скільки зароблено монет за 4 часовий бонус до першої покупки;
54. time_bonus_collect_total – скільки зароблено монет за 4 часовий бонус загалом;
55. torch_used – скільки використано факелів до першої покупки;
56. torch_used_total – скільки використано факелів загалом;

57. total_levels_24h – скільки рівнів було пройдено за перші 24 години;
58. total_payments – скільки всього покупок було зроблено;
59. video_reward_collect – скільки зароблено монет за перегляд реклами до першої покупки;
60. video_reward_collect_total – скільки зароблено монет за перегляд реклами загалом;
61. win – скільки всього перемог.

2.1.2 Співвідношення даних за класом, перевірка на дублікати, заповнення пропусків та виявлення дефектних значень.

Співвідношення за головним класом можна побачити на рис. 3. Незбалансованість є, але різниця не така велика, тому при навчанні буде задіяний метод, коли менш збалансованому класу буде задані ваги, які дорівнюють долі більш збалансованого класу. Тобто у даному випадку це 0.7. Навпаки, для більш збалансованого класу будуть призначенні ваги 0.3.

```
0    0.690749
1    0.309251
Name: is_paying, dtype: float64
```

Рис. 3 – співвідношення за головним класом

Дублікати не були виявлені (рис. 4).

```
print(len(df[df.duplicated()]), 'length of duplicates')
```

```
0 length of duplicates
```

Рис. 4 – відсутність дублікатів

Переважну кількість стовпців з пропущеними значеннями були заповнені за принципом генерації значення між 40% та 60% квантилем. video_reward_collect, video_reward_collect_total, level_failed_missclick, level_failed_quit та level_failed_time були заповнені нулями. А стовпець total_levels_24h був прибраний із датасету, тому що було пропущено дуже багато

значень саме для класу, який відноситься до платячих. Тільки 42 гравці мають непропущене значення. Навіть при такому співвідношенні видно, що гравці, які платять за перші 24 години проходять більше рівнів ніж ті, хто не платить (рис. 5). Тому потенційно це сильний індикатор.

is_paying	count	mean	median
0	1023	20.182796	18.0
1	42	30.404762	23.5

Рис. 5 – зведена таблиця за total_levels_24h та is_paying

Також за допомогою «ящика з вусами» було виявлено кілька дефектних значень к стовпці avg_win_rate. Там деякі значення дорівнювали -1 (рис. 6).

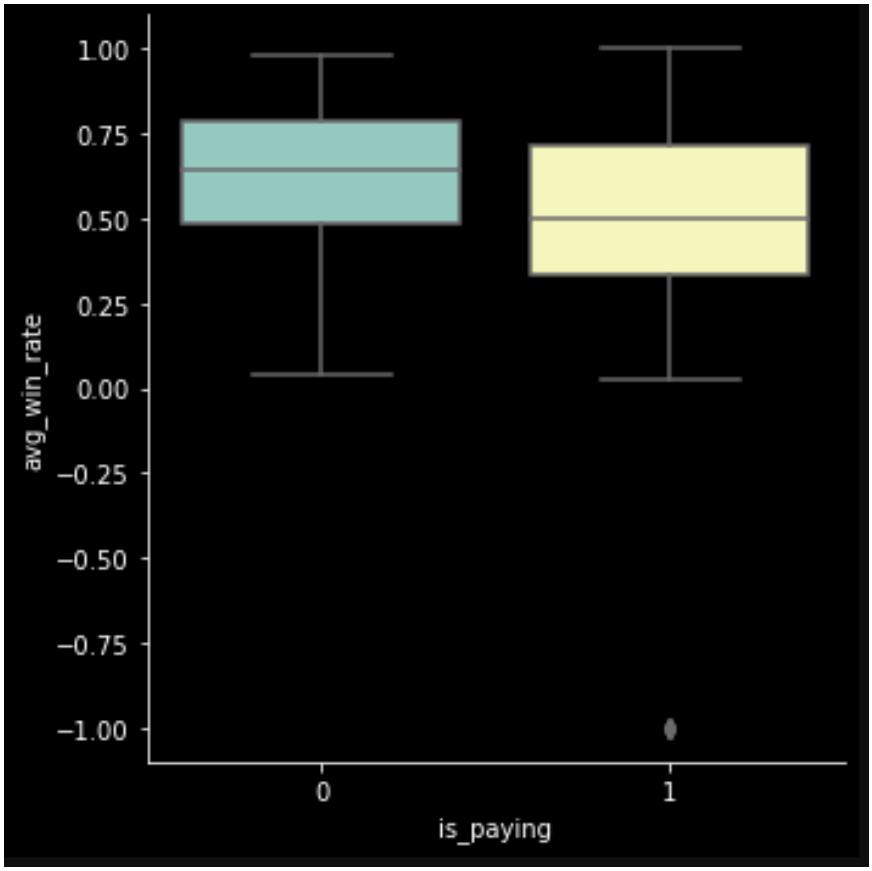

Рис. 6 – виявлення дефектних значень

2.1.3 Масштабування значень до контексту одного пройденого рівня

Щоб можна було адекватно оцінювати гравців, треба перевести заробіток та витрачання ресурсів в контекст одного пройденого рівня. Це було зроблено за допомогою стовпців day_levels_passed_total та day_levels_passed.

2.1.4 Розвідувальний аналіз даних

Спочатку було звернуто увагу на те, скільки платячи гравці використовують бонусів за один рівень до та після покупки (рис. 7). Чітко видно, що при більшій наявності ресурсів більше застосовуються бонуси.

```
print('Torch')
print(df_scaled[df_scaled['is_paying'] == 1]['torch_used'].mean())
print(df_scaled[df_scaled['is_paying'] == 1]['torch_used_total'].mean())
```

```
Torch
0.37275346470147525
0.48740953707535123
```

```
print('Hint')
print(df_scaled[df_scaled['is_paying'] == 1]['hint_used'].mean())
print(df_scaled[df_scaled['is_paying'] == 1]['hint_used_total'].mean())
```

```
Hint
0.9249183389013311
1.1318781005478045
```

```
print('Bomb')
print(df_scaled[df_scaled['is_paying'] == 1]['bomb_used'].mean())
print(df_scaled[df_scaled['is_paying'] == 1]['bomb_used_total'].mean())
```

```
Bomb
0.6203242176688203
0.7670801886481251
```

```
print('Freeze')
print(df_scaled[df_scaled['is_paying'] == 1]['freeze_used'].mean())
print(df_scaled[df_scaled['is_paying'] == 1]['freeze_used_total'].mean())
```

```
Freeze
0.46684267122518397
0.620348593957801
```

```
print('Extra time')
print(df_scaled[df_scaled['is_paying'] == 1]['extra_time_used'].mean())
print(df_scaled[df_scaled['is_paying'] == 1]['extra_time_used_total'].mean())
```

```
Extra time
0.3436678610739673
0.46220704355843756
```

Рис. 7 – використання бонусів платячими гравцями до та після першої покупки

Він-рейт та кількість пройдений рівнів за день теж не змінюється (рис. 8).

```
8.8668903803132 avg_day_levels_passed
8.435123042505593 avg_day_levels_passed_total
```

```
0.5310153065061729 avg_win_rate
0.502678953515469 4 avg_win_rate_total
```

Рис. 8 – він-рейт та кількість пройдених рівнів до та після першої покупки

Далі, було звернуто увагу на гендер гравця, але там було дуже багато невизначених полів. Тому було вирішено позбутися цього стовпця для навчання. Але навіть при такій ситуації видно, що жінки трішки більше купують ніж чоловіки (рис. 9).

	count	mean
gender		
No_info	652	0.638037
andy	7	0.000000
female	518	0.042471
male	122	0.016393
mostly_female	25	0.160000
mostly_male	7	0.000000
unknown	139	0.021583

Рис. 9 – зведена таблиця за gender та is_paying

Далі були розглянуті наявні ресурси у гравців на цей момент. Видно, що ті хто роблять покупки, мають більше ресурсів з перевагою (рис. 10). Логічно, що

гравці які купують ігрові ресурси будуть мати більше їх в сумі. Тому при побудові моделі ці характеристики не потрібні.

	current_bomb		current_energy		current_freeze	
	count	mean	count	mean	count	mean
is_paying						
0	1023	0.070759	1023	0.060529	1023	0.082819
1	447	0.182158	447	0.175951	447	0.549555

	current_hint		current_money		current_torch	
	count	mean	count	mean	count	mean
is_paying						
0	1023	0.067595	1023	247.869929	1023	0.084182
1	447	0.177641	447	920.324391	447	0.369603

Рис. 10 – Зведені таблиці за is_paying та наявними ресурсами.

Далі були розглянуті гравці за браузером. За зведеною таблицею на рис. 11 можна сказати, що браузер не впливає на is_paying, тому ця змінна не буде використана при навчанні.

	is_paying	
	count	mean
browser_name		
Chrome	1176	0.307823
Firefox	234	0.290598
Netscape	1	0.000000
Opera	27	0.185185
Safari	32	0.375000

Рис. 11 – зведена таблиця за is_paying та browser_name

Для більш зручної роботи всі стовпці, що відповідають за заробіток та витрати були об’єднані в окремі стовпці money_collect, money_collect_total, money_remove, money_remove_total.

При аналізі програшів рівнів було виявлено, що платячі гравці частіше програють ніж неплатячі, а саме через час (рис. 12).

	level_failed_time		level_failed_total	
	count	mean	count	mean
is_paying				
0	1023	0.571400	1023	0.952828
1	447	1.526651	447	1.870159

Рис. 12 – демонстрація того, що неплатячі гравці частіше програють через час

Далі була розглянута змінна new_user_entry_point, але вона також особливо не впливає на клас гравця (рис. 13).

	is_paying	
	count	mean
new_user_entry_point		
admin_message	6	0.500000
bookmark	3	0.666667
facebook_gaming_tab	10	0.300000
facebook_web	14	0.357143
failed_to_get_entry_point	10	1.000000
feed	3	1.000000
game_switch	168	0.410714
in_game_menu	557	0.292639
other	1	0.000000
shareable_link	5	0.400000
web_games_hub	693	0.269841

Рис. 13 – зведена таблиця за new_user_entry_point та is_paying

Було вирішено виражати трати та зароблені монети через коефіцієнт (рис. 14) money_ratio, який вираховувався співвідношенням зароблених монет до витрачених монет. Якщо, money_ratio < 1, тоді гравець витрачає більше ніж заробляє.

Якщо, money_ratio > 1, тоді гравець отримує більше ніж витрачає.

Треба звернути увагу на тих, у кого coef > 1 і він неплатячий.

```
count    1470.000000
mean        1.343136
std         1.324668
min         0.027502
25%         0.778877
50%         0.997837
75%         1.448040
max        18.357488
Name: money_ratio, dtype: float64
```

Рис. 14 – загальна інформація про коефіцієнт трат та заробітку

Була побудована зведена таблиця за money_ratio та is_paying (рис. 15) на якій видно, що за медіаною неплатячі гравці мають цей коефіцієнт біля 1, а платячі біля 0.75. Це означає, що неплатячим вдається витрачати та заробляти однаково, тому у них і нема потреби в докупках, а платячим завжди не вистачає, тому вони і докуповують.

	money_ratio		
	count	mean	median
is_paying			
0	1023	1.470912	1.069035
1	447	1.050710	0.745914

Рис. 15 – зведена таблиця за money_ratio та is_paying

Тепер коли характеристик залишилось не так багато, можна подивитися на коефіцієнт phik у вигляді таблиці (рис. 16). Видно, що на is_paying мають вплив win_rate, booster_used та day_passed. Money_ratio за цим коефіцієнтом впливу не має.

	day_passed	level_time	win_rate	first_day_passed	is_paying	booster_used	money_ratio
day_passed	1.000000	0.000000	0.173442	0.778712	0.250960	0.000000	0.000000
level_time	0.000000	1.000000	0.110847	0.245903	0.091634	0.436577	0.000000
win_rate	0.173442	0.110847	1.000000	0.072987	0.337736	0.389268	0.190481
first_day_passed	0.778712	0.245903	0.072987	1.000000	0.244805	0.000000	0.000000
is_paying	0.250960	0.091634	0.337736	0.244805	1.000000	0.316703	0.039254
booster_used	0.000000	0.436577	0.389268	0.000000	0.316703	1.000000	0.000000
money_ratio	0.000000	0.000000	0.190481	0.000000	0.039254	0.000000	1.000000

Рис. 16 – вираховування коефіцієнту phik

Побудувавши три графічні представлення (рис. 17, жовтий – це платячі, зелений - неплатячі) scatter plot за day_passed, win_rate, money_ratio та booster_used та 3D графік за booster_used, money_ratio та win_rate видно, що класи гравців на фоні цих характеристик можна розрізняти. Платячі використовують більше бонусів та більше грають.

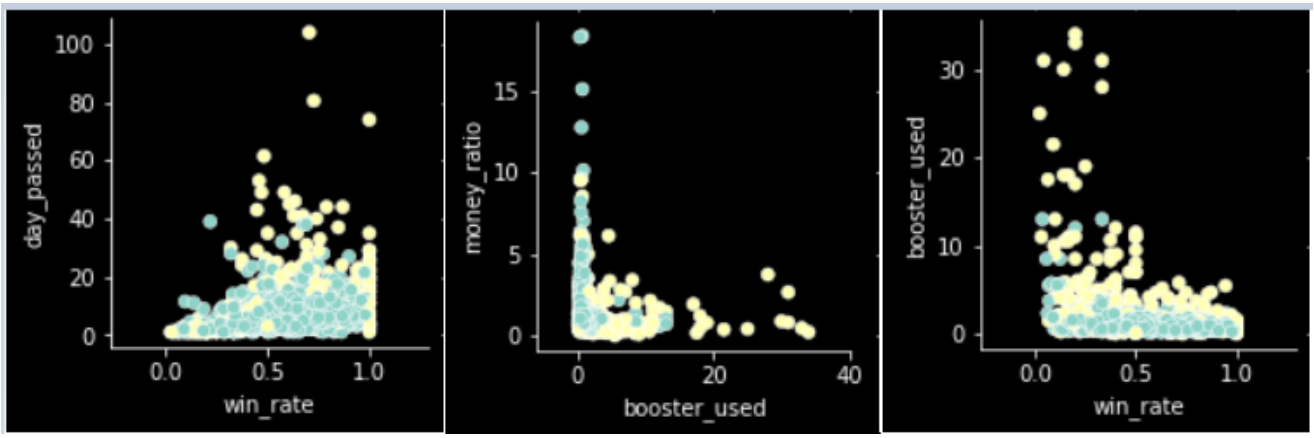

Рис. 17 – scatter plot за day_passed, win_rate, money_ratio та booster_used

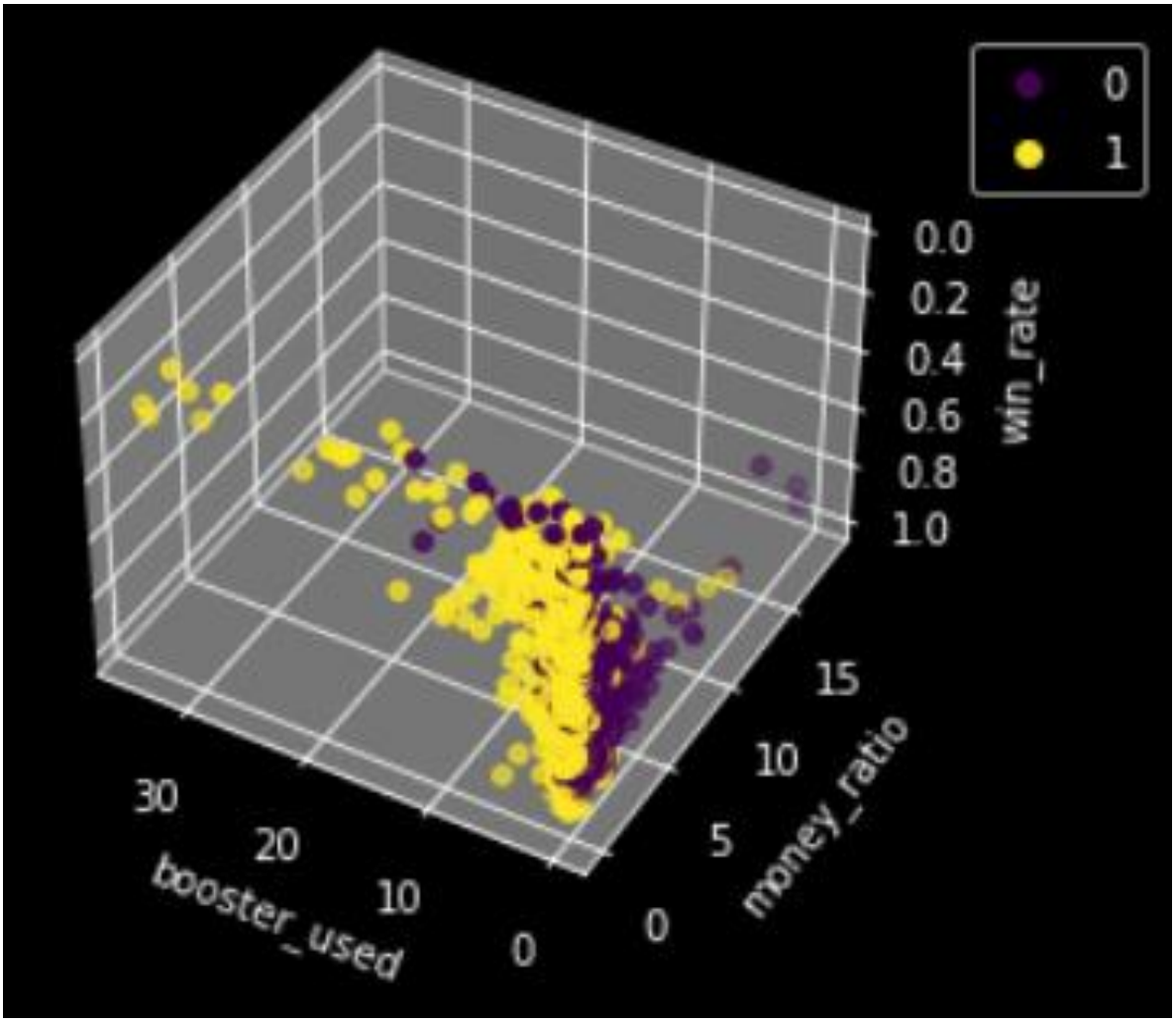

Рис. 18 – 3D графік за booster_used, money_ratio та win_rate

2.1.5 Перевірка на мультиколінеарність

Для перевірки датасету на мультиколінеарність був використаний коефіцієнт VIF (рис. 19). Так як усі значення менше 10, то мультиколінеарність відсутня.

	feature	VIF
2	win_rate	7.086906
1	level_time	5.749873
4	money_ratio	2.164609
0	day_passed	2.138594
3	booster_used	1.452629

Рис. 19 – значення VIF

2.1.6. Розбиття на тренувальну та тестові вибірки

Розміри тренувальної та тестової вибірок можна побачити на рис. 20.

```
y_train.value_counts()
0    712
1    317
Name: is_paying, dtype: int64
```

```
y_test.value_counts()
0    311
1    130
Name: is_paying, dtype: int64
```

Рис. 20 – розміри тренувальної та тестової вибірок

Також за допомогою «ящика з вусами» були відсічені викиди для тренувальної вибірки.

2.1.7 Масштабування

Для масштабування значень був застосований MinMaxScaler, який переведе всі значення в діапазон від 0 до 1.

2.1.8. Висновок перед навчанням

Люди, які не платять краще грають ніж платячі. Це можуть бути фактори різні, такі як: реакція, зір, вік та інше. Видно, що неплатячі проходять в середньому в день стільки ж скільки платячі, при цьому вони набагато менше тратять бонусів, у них більше він-рейт та вони більше заробляють ніж витрачають, просто тому що вони мають кращі навички гри. За даних умов гри їм і не потрібно робити покупку, тому що вони отримують в середньому в день

стільки ж контенту, скільки отримують платячі гравці, застосовуючи більше зусиль та витрачаючи купу бонусів. Виходом із даної ситуації може бути ускладнення гри конкретно для неплатячих гравців, а для платячих гравців залишити все так як є. На рис. 21 зображена загальна зведена таблиця за таргет-класом та характеристиками.

	booster_used			money_ratio			win_rate		
	count	mean	median	count	mean	median	count	mean	median
is_paying									
0	1023	0.971338	0.789474	1023	1.470912	1.069035	1023	0.622926	0.641509
1	447	2.728507	1.333333	447	1.050710	0.745914	447	0.531015	0.500000

	level_time			day_passed		
	count	mean	median	count	mean	median
is_paying						
0	1023	69.006354	67.0	1023	6.825024	5.5
1	447	74.200224	66.0	447	8.866890	5.0

Рис. 21 – загальна зведена таблиця за таргет-класом та характеристиками

2.2 Задача класифікації

2.2.1 Random Forest

За методом GridSearchCV були виявлені оптимальні гіперпараметри, які можна побачити на рис. 22.

```
{'bootstrap': True,
 'class_weight': {0: 0.3, 1: 0.7},
 'max_depth': 5,
 'max_features': 3,
 'min_samples_leaf': 5,
 'min_samples_split': 5,
 'n_estimators': 200,
 'oob_score': True}
```

Рис. 22 – гіперпараметри Random Forest

На рис. 23 зображений звіт про класифікацію на тренувальних даних, де можна побачити precision, recall, f1-score та accuracy.

	precision	recall	f1-score	support
0	0.92	0.84	0.88	707
1	0.69	0.82	0.75	297
accuracy			0.84	1004
macro avg	0.80	0.83	0.81	1004
weighted avg	0.85	0.84	0.84	1004

Рис. 23 – звіт про класифікацію на тренувальних даних

На рис. 24 зображений Confusion Matrix для тренувальних даних, де можна побачити скільки разів модель правильно розпізнала класи і скільки разів помилилася.

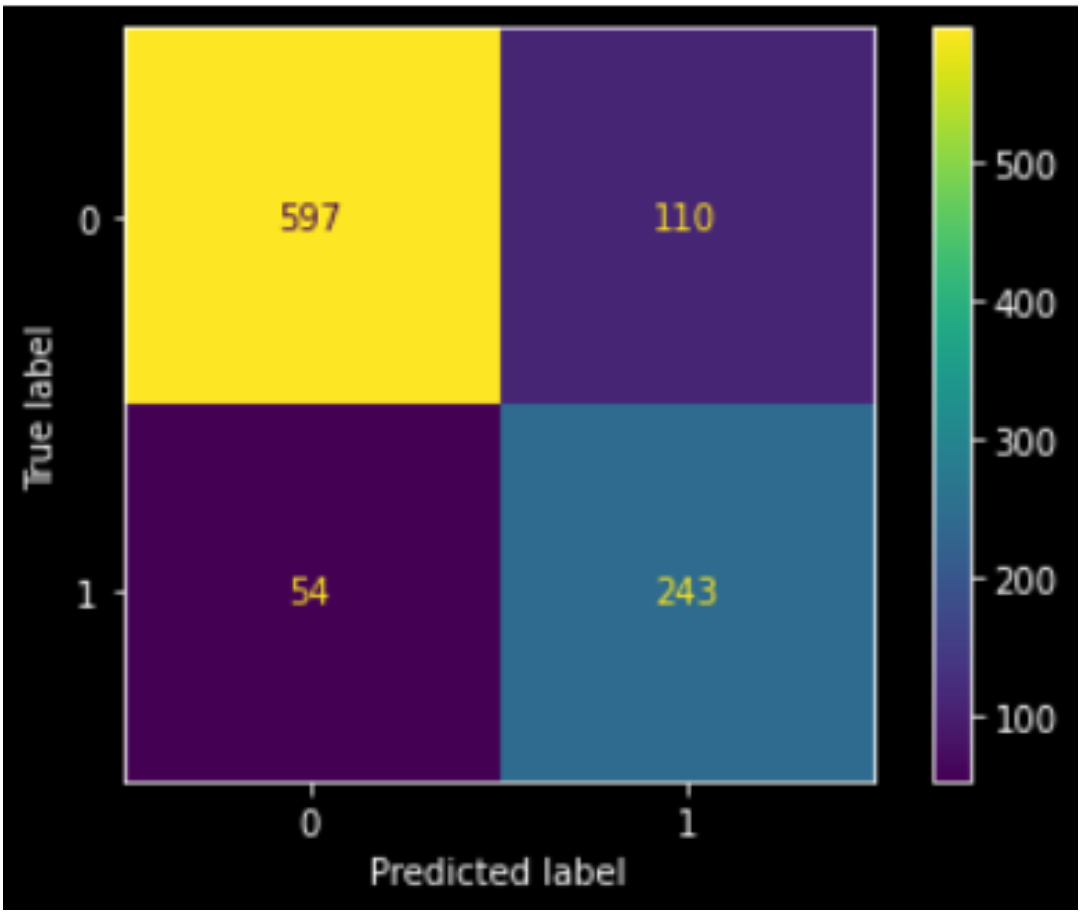

Рис. 24 – Confusion Matrix для тренувальних даних

На рис. 25 зображена Крива ROC та AUC для тренувальних даних. AUC score дорівнює 0.83.

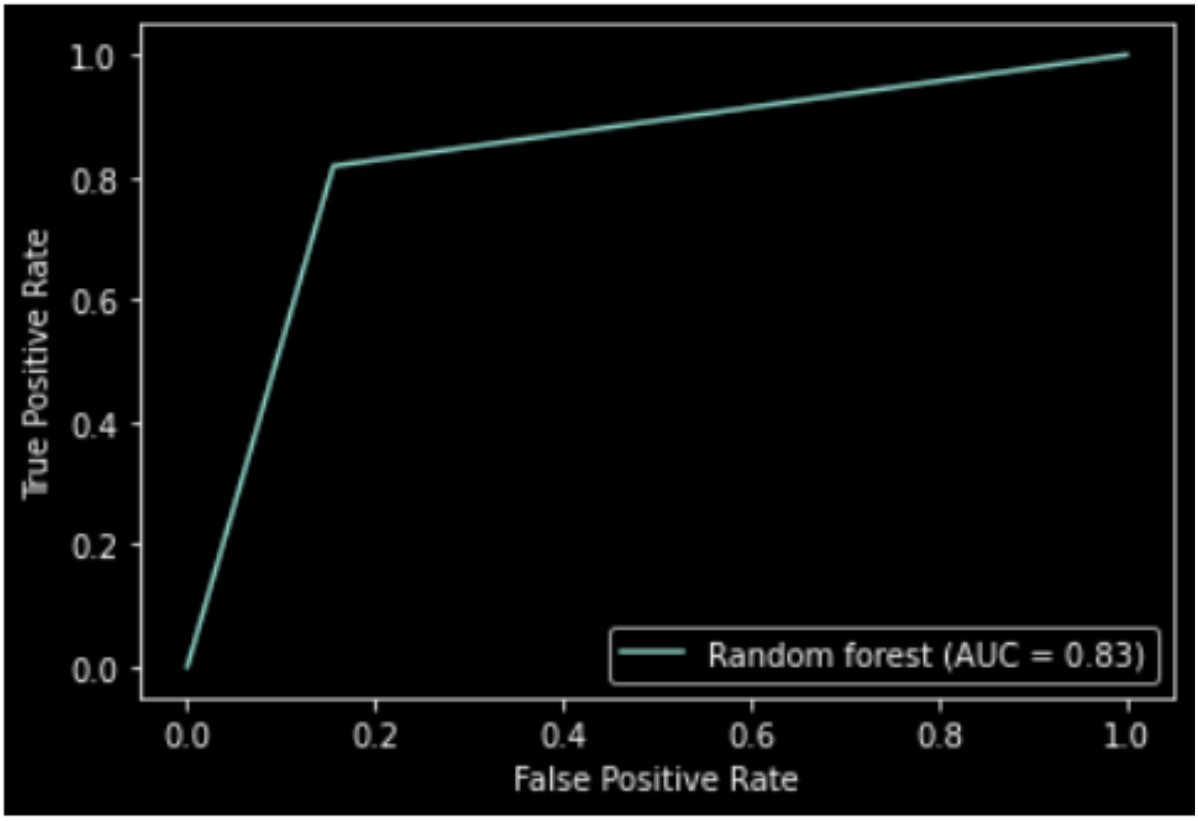

Рис. 25 – Крива ROC та AUC для тренувальних даних

На рис. 26 зображена важливість характеристик на визначення класу. Найбільш сильною характеристикою є money_ratio. Також немалу роль грають booster_used та win_rate. Зовсім малу роль грають day_passed та level_time.

	importance
money_ratio	0.495836
booster_used	0.255309
win_rate	0.135400
day_passed	0.057061
level_time	0.056395

Рис. 26 – Важливість характеристик на визначення класу

На рис. 27 зображений звіт про класифікацію на тестових даних, де можна побачити precision, recall, f1-score та accuracy.

	precision	recall	f1-score	support
0	0.91	0.80	0.85	311
1	0.63	0.80	0.70	130
accuracy			0.80	441
macro avg	0.77	0.80	0.78	441
weighted avg	0.82	0.80	0.81	441

Рис. 27 - звіт про класифікацію на тестувальних даних

На рис. 28 зображений Confusion Matrix для тестових даних, де можна побачити скільки разів модель правильно розпізнала класи і скільки разів помилилася.

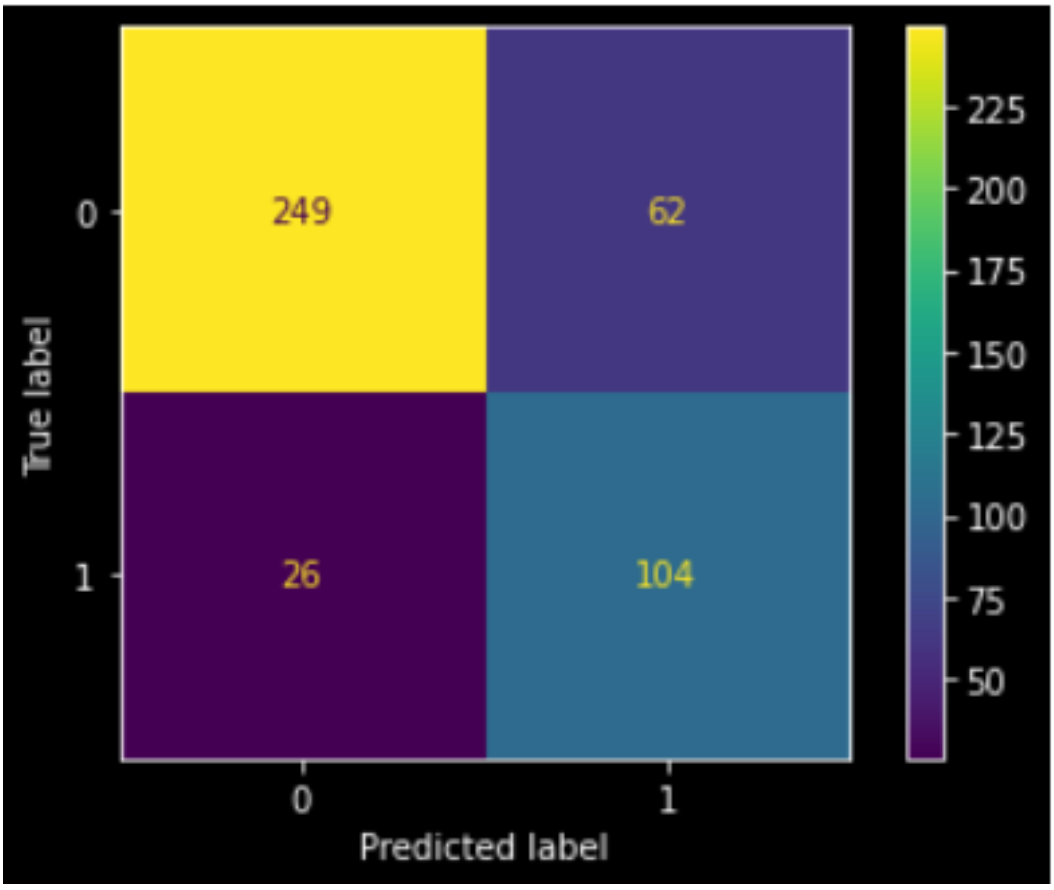

Рис. 28 - Confusion Matrix для тестових даних

На рис. 29 зображена Крива ROC та AUC для тестових даних. AUC score дорівнює 0.8.

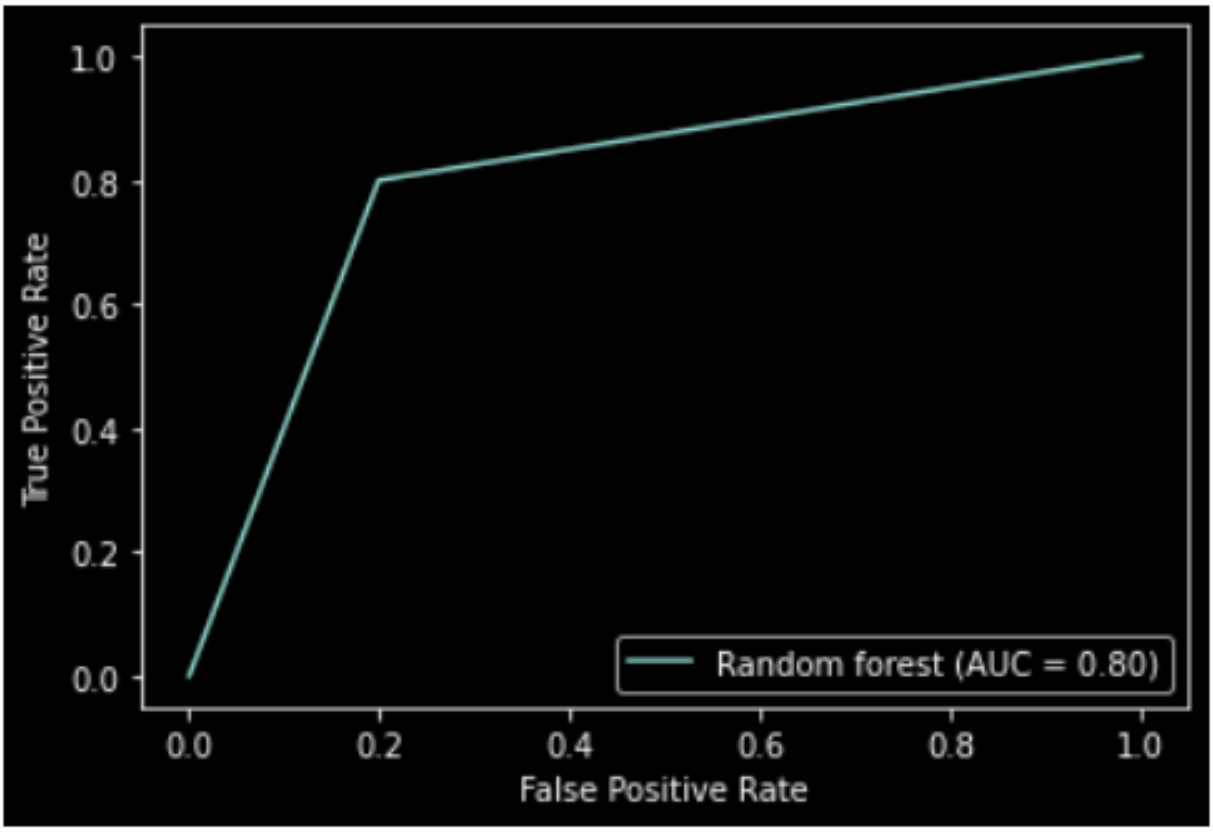

Рис. 29 - Крива ROC та AUC для тестових даних

2.2.2 Logistic Regression

За методом GridSearchCV були виявлені оптимальні гіперпараметри, які можна побачити на рис. 30.

```
{
'C': 2.7825594022071245,
'class_weight': {0: 0.3, 1: 0.7},
'penalty': 'l1'
}
```

Рис. 30 – гіперпараметри Logistic Regression

На рис. 31 зображений звіт про класифікацію на тренувальних даних, де можна побачити precision, recall, f1-score та accuracy.

	precision	recall	f1-score	support
0	0.87	0.52	0.65	707
1	0.41	0.81	0.55	297
accuracy			0.60	1004
macro avg	0.64	0.67	0.60	1004
weighted avg	0.73	0.60	0.62	1004

Рис. 31 – звіт про класифікацію на тренувальних даних

На рис. 32 зображений Confusion Matrix для тренувальних даних, де можна побачити скільки разів модель правильно розпізнала класи і скільки разів помилилася.

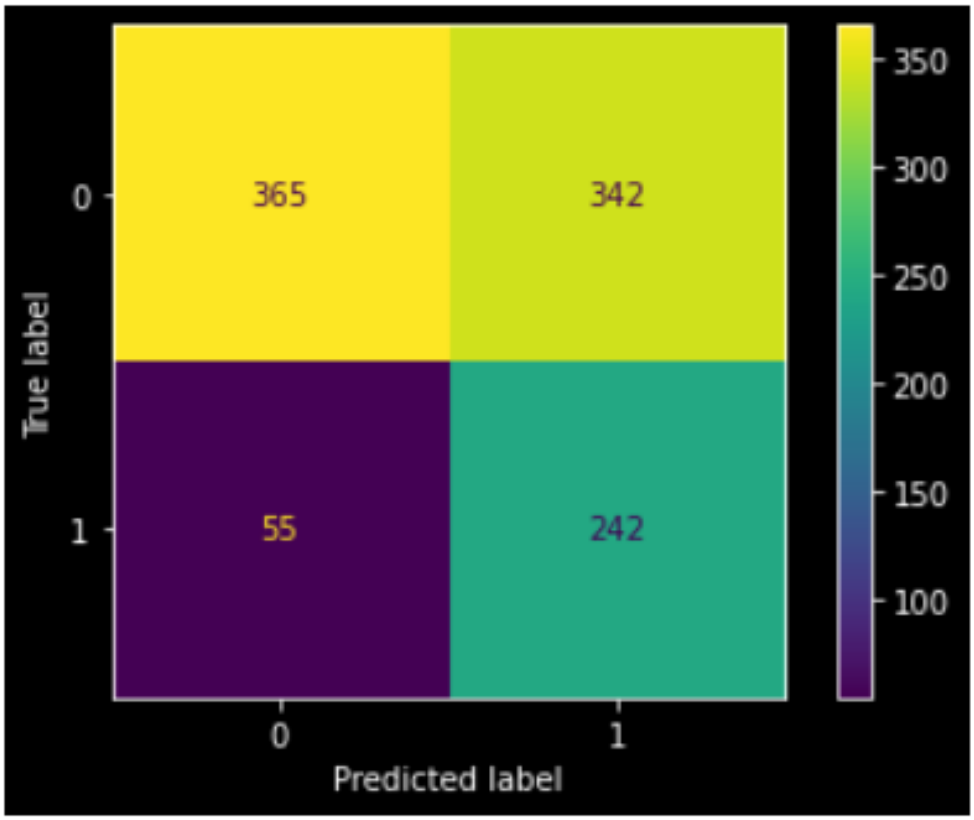

Рис. 32 – Confusion Matrix для тренувальних даних

На рис. 33 зображена Крива ROC та AUC для тренувальних даних. AUC score дорівнює 0.67.

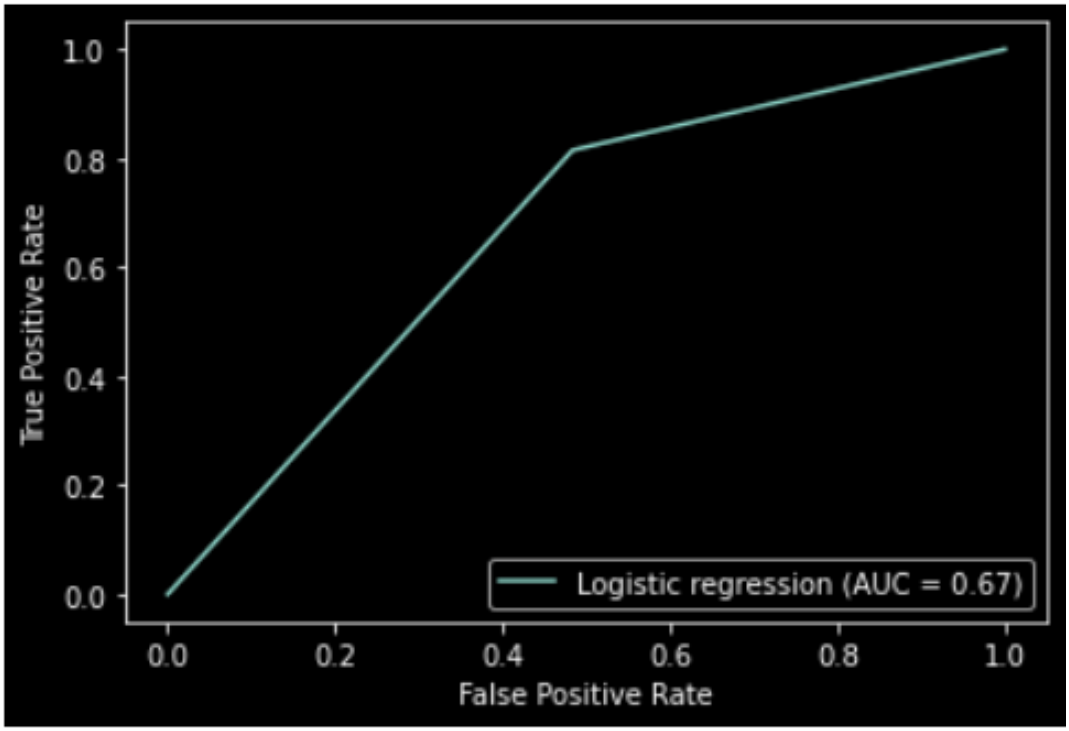

Рис. 33 – Крива ROC та AUC для тренувальних даних

На рис. 34 зображена важливість характеристик на визначення класу. Найбільш сильною характеристикою є booster_used. Також немалу роль грає money_ratio. Зовсім малу роль грають day_passed, level_time та win_rate.

	Coefs
money_ratio	-2.992242
win_rate	-0.590485
level_time	-0.105422
day_passed	3.589131
booster_used	10.433791

Рис. 34 – Важливість характеристик на визначення класу

На рис. 35 зображений звіт про класифікацію на тестових даних, де можна побачити precision, recall, f1-score та accuracy.

	precision	recall	f1-score	support
0	0.88	0.53	0.67	311
1	0.43	0.83	0.56	130
accuracy			0.62	441
macro avg	0.65	0.68	0.61	441
weighted avg	0.75	0.62	0.64	441

Рис. 35 - звіт про класифікацію на тестувальних даних

На рис. 36 зображений Confusion Matrix для тестових даних, де можна побачити скільки разів модель правильно розпізнала класи і скільки разів помилилася.

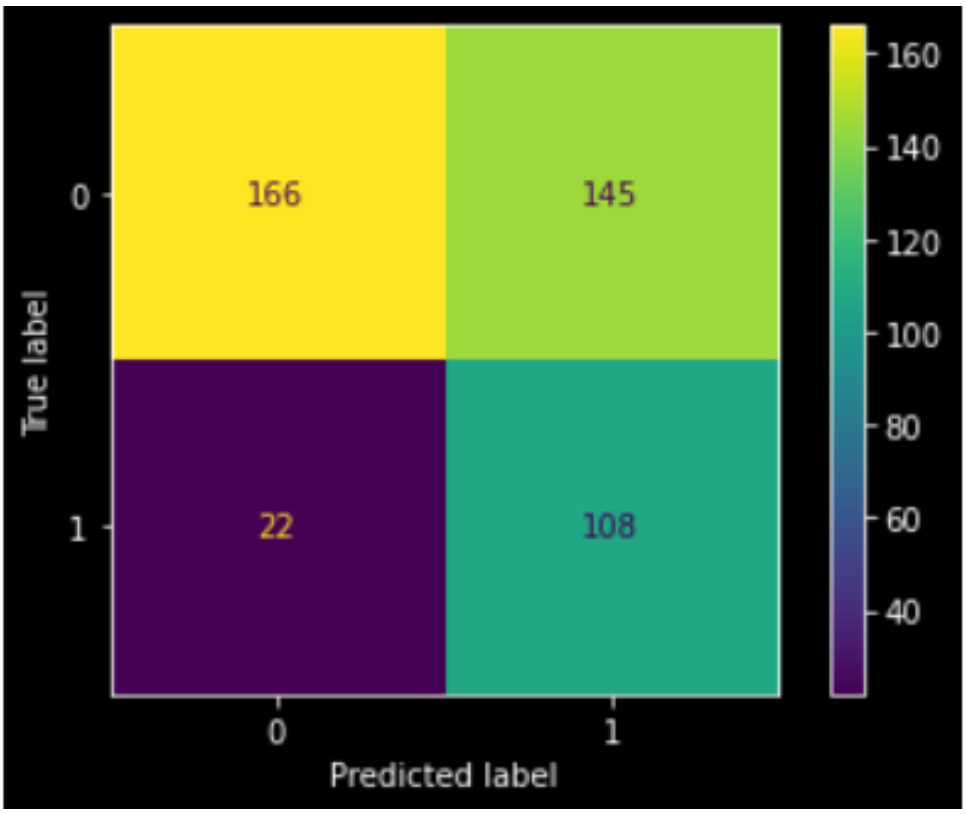

Рис. 36 - Confusion Matrix для тестових даних

На рис. 37 зображена Крива ROC та AUC для тестових даних. AUC score дорівнює 0.8.

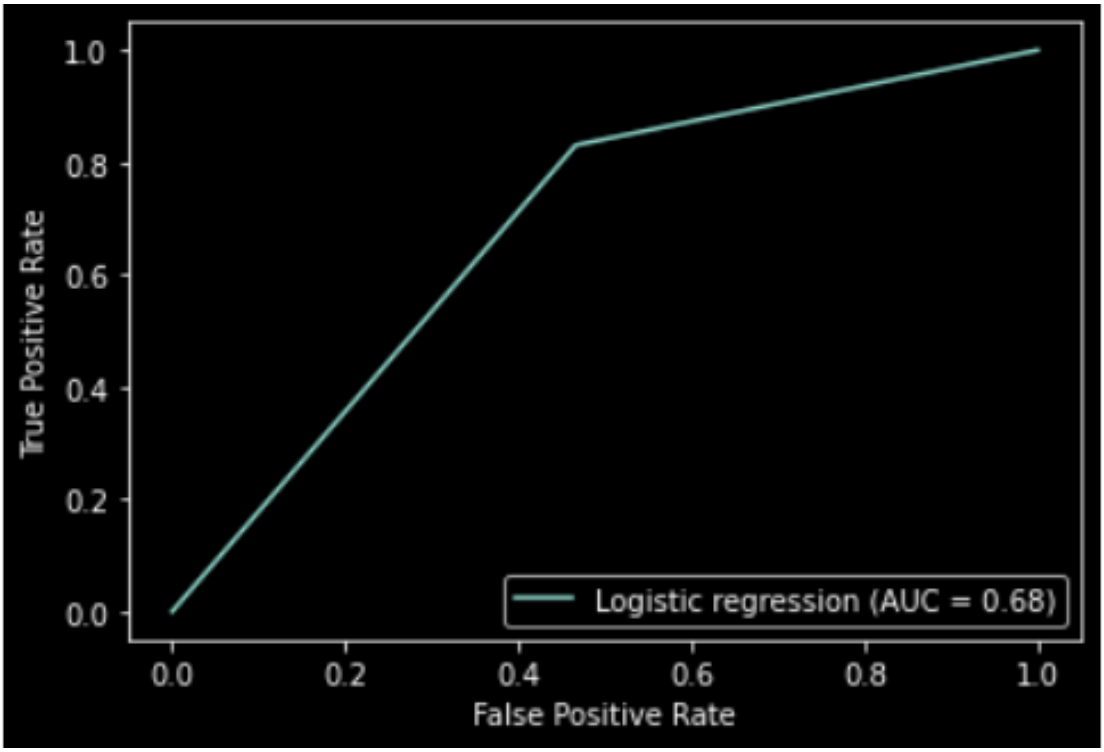

Рис. 37 - Крива ROC та AUC для тестових даних

2.2.3 Support Vector Machines

За методом GridSearchCV були виявлені оптимальні гіперпараметри, які можна побачити на рис. 38.

```
{'C': 0.1,
 'class_weight': {0: 0.3, 1: 0.7},
 'degree': 3,
 'gamma': 'scale',
 'kernel': 'poly',
 'probability': True}
```

Рис. 38 – гіперпараметри Logistic Regression

На рис. 39 зображений звіт про класифікацію на тренувальних даних, де можна побачити precision, recall, f1-score та accuracy.

	precision	recall	f1-score	support
0	0.80	0.84	0.82	707
1	0.57	0.51	0.54	297
accuracy			0.74	1004
macro avg	0.68	0.67	0.68	1004
weighted avg	0.73	0.74	0.74	1004

Рис. 39 – звіт про класифікацію на тренувальних даних

На рис. 40 зображений Confusion Matrix для тренувальних даних, де можна побачити скільки разів модель правильно розпізнала класи і скільки разів помилилася.

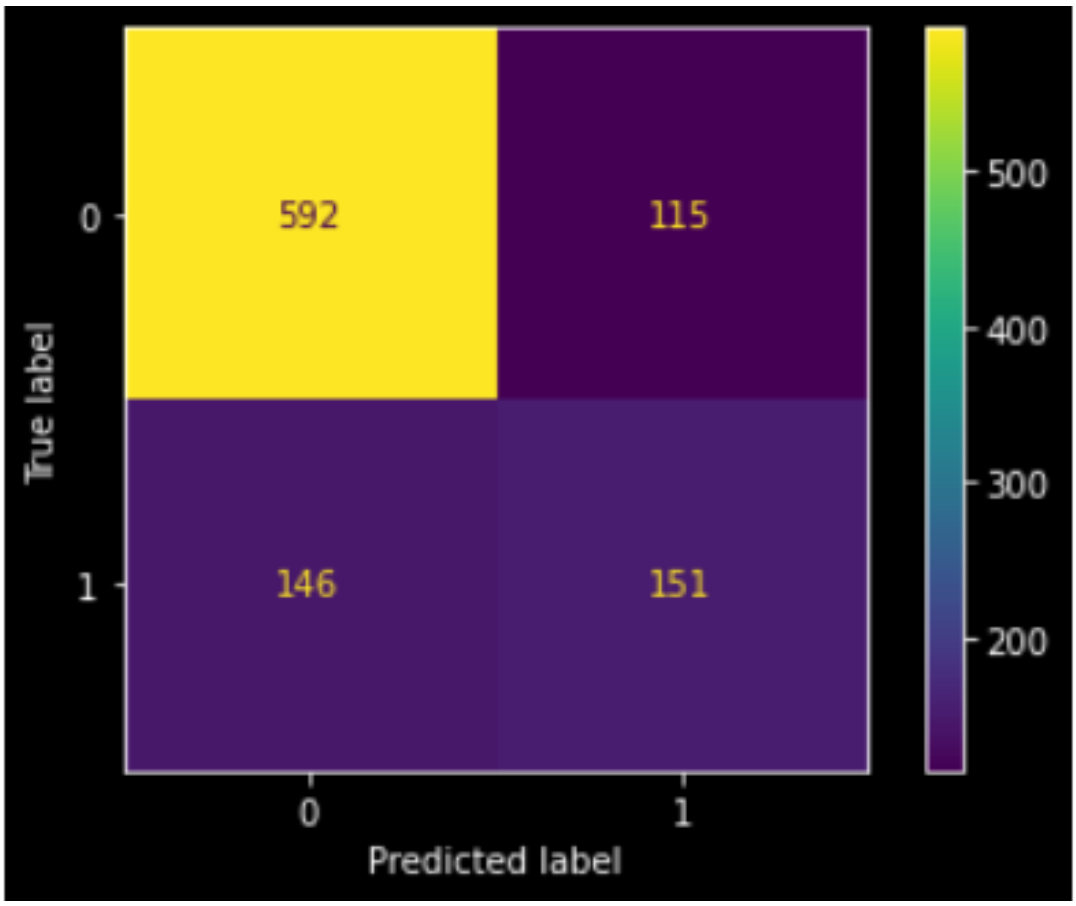

Рис. 40 – Confusion Matrix для тренувальних даних

На рис. 41 зображена Крива ROC та AUC для тренувальних даних. AUC score дорівнює 0.67.

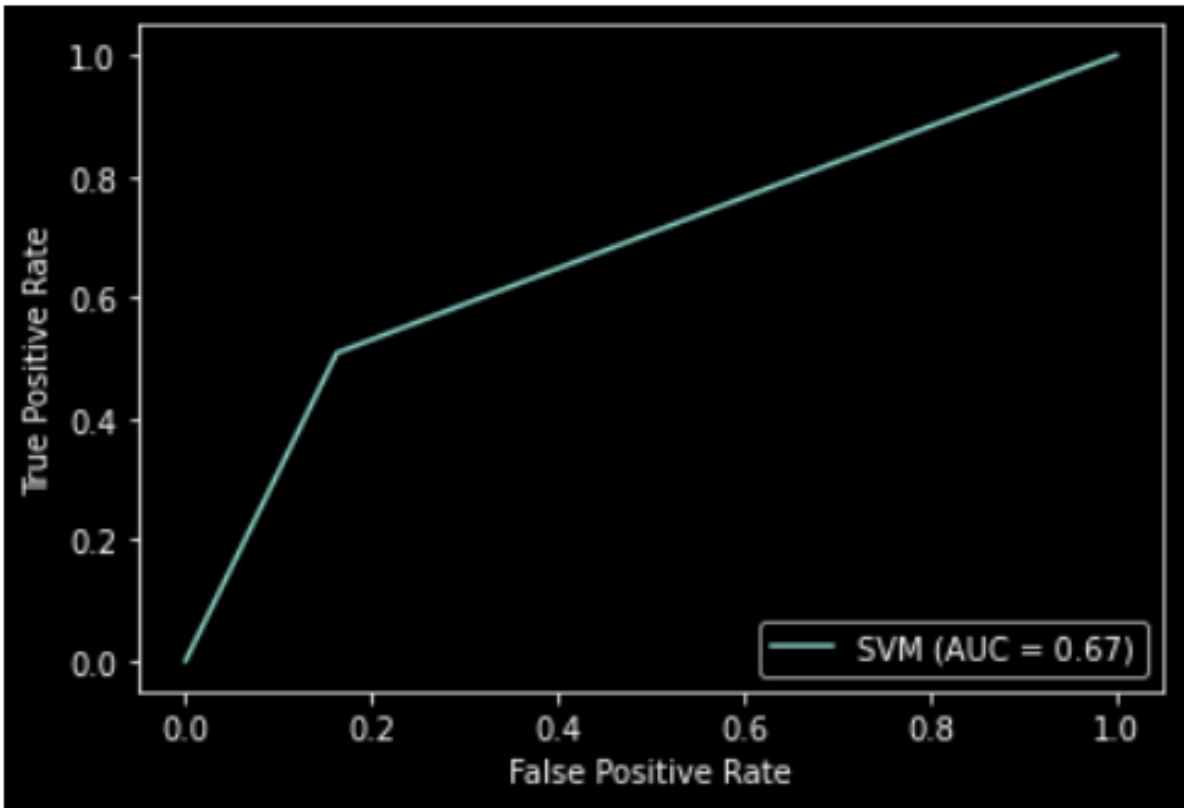

Рис. 41 – Крива ROC та AUC для тренувальних даних

На рис. 42 зображений звіт про класифікацію на тестових даних, де можна побачити precision, recall, f1-score та accuracy.

	precision	recall	f1-score	support
0	0.81	0.84	0.82	311
1	0.58	0.54	0.56	130
accuracy			0.75	441
macro avg	0.70	0.69	0.69	441
weighted avg	0.74	0.75	0.75	441

Рис. 42 - звіт про класифікацію на тестувальних даних

На рис. 43 зображений Confusion Matrix для тестових даних, де можна побачити скільки разів модель правильно розпізнала класи і скільки разів помилилася.

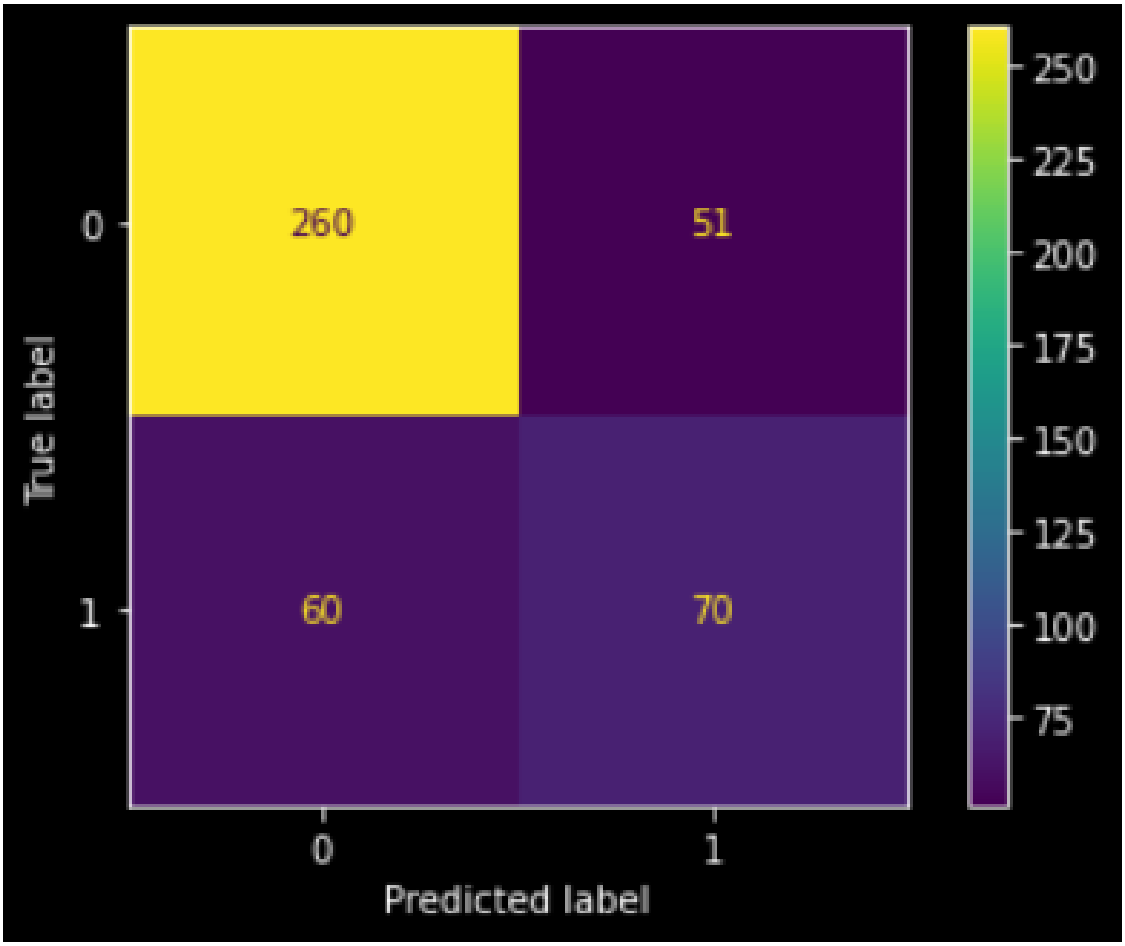

Рис. 43 - Confusion Matrix для тестових даних

На рис. 44 зображена Крива ROC та AUC для тестових даних. AUC score дорівнює 0.69.

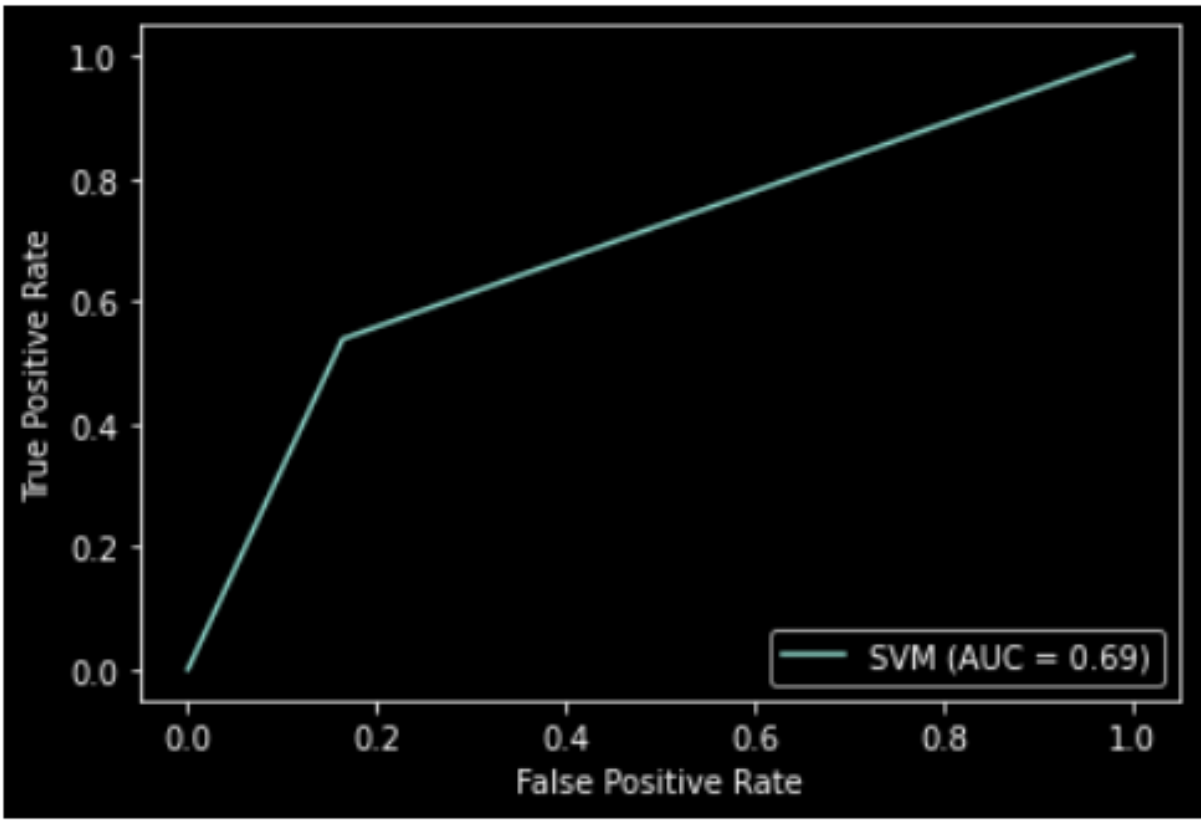

Рис. 44 - Крива ROC та AUC для тестових даних

2.2.4 Вибір моделі класифікації

В таблиці 1 наведені показники ефективності для трьох алгоритмів класифікації, які були розглянуті в цій роботі. Опираючись на них можна зробити висновок, що алгоритм Random Forest відпрацював найліпше, а Logistic Regression та Support Vector Machines не пройшли поріг метрики AUC, що дорівнює 0.7. Тобто, вирішити задачу класифікації вдалося тільки за алгоритмом Random Forest.

Отже,

	AUC score	**Accuracy**	**F1-score (1)**	**F1-score (0)**
Random Forest	0.8	0.8	0.7	0.85
Logistic Regression	0.68	0.62	0.56	0.67
Support Vector Machines	0.69	0.75	0.56	0.82

Табл. 1 – Показники ефективності алгоритмів класифікації

2.3 Кластеризація

Для більш детального налаштування складності гри для неплатячих гравців було вирішено провести кластерний аналіз саме для неплатячих гравців за допомогою алгоритму K-means.

2.3.1 K-means

Для скейлінгу значень був використаний алгоритм MinMaxScaler. Далі було обрано кількість кластерів за допомогою графіка кам’янистої осипі рис. 45. Було вирішено обрати кількість кластерів n = 3.

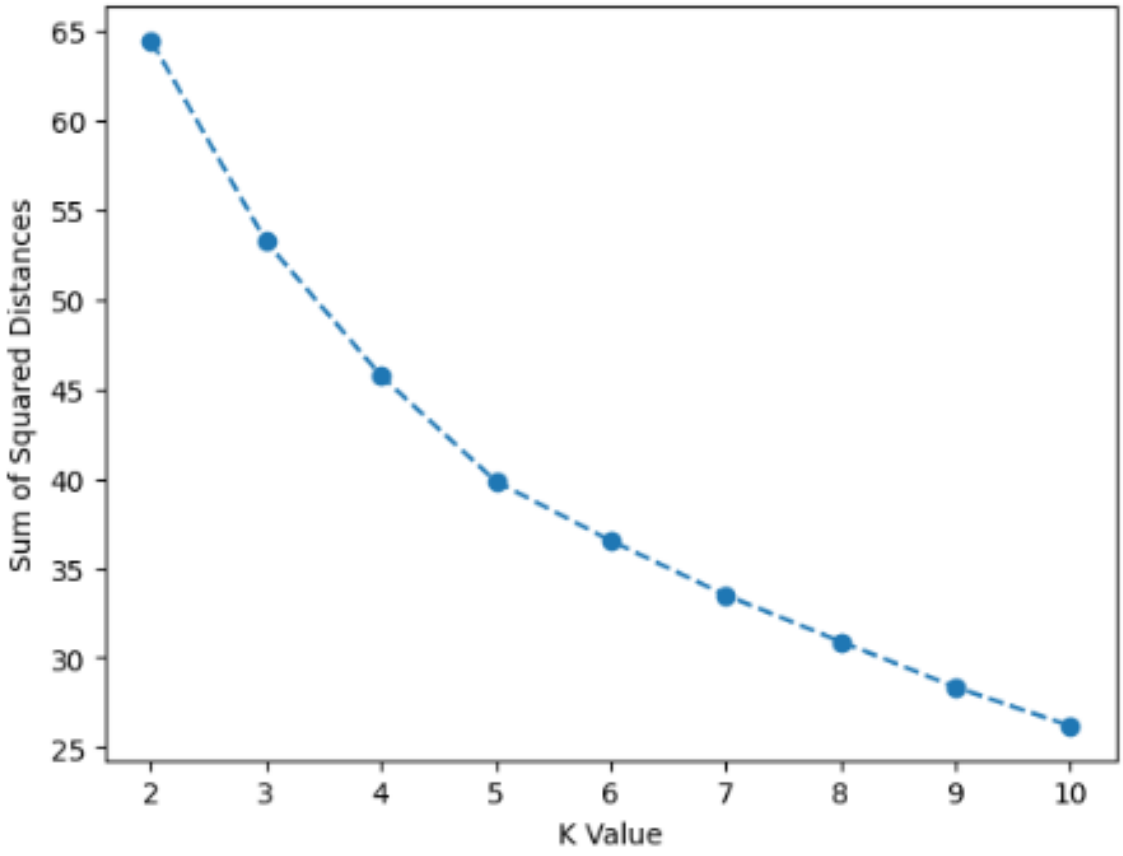

Рис. 45 – Графік кам’янистої осипі

Після визначенні кількості кластерів була проведена їх інтерпретація. На рис. 46 зображена зведена таблиця кластерів з усередненими значеннями.

cluster	0	1	2
day_passed	4.000000	7.500000	4.000000
level_time	93.000000	62.000000	67.000000
win_rate	0.631579	0.783196	0.440678
is_paying	0.000000	0.000000	0.000000
booster_used	1.000000	0.666667	0.876362
money_ratio	1.038870	1.303003	0.981129

Рис. 46 – зведена таблиця кластерів з усередненими значеннями

Інтерпретація груп:

0 – люди, які дуже виділяються на тлі інших у плані часу витраченого на рівень. Але при цьому примудряються збирати більше монет ніж витрачати, при тому що часто беруть бустери. І мають непоганий він-рейт. Для таких гравців є сенс ускладнення гри саме збільшенням предметів на рівні.

1 – люди, які добре грають, мають величезний він-рейт. Використовують мало бустерів та швидко проходять. І мають величезний надлишок монет. І головне їм подобається гра. Проходять 7.5 рівнів на день. Можна вважати, що це головні кандидати на скорочення часу.

2 – люди, які не так добре грають. Мають невеликий він-рейт. Часто використовують бустери. Витрачають майже стільки ж скільки заробляють. Можна припустити, що ці не заплатять ні в якому разі, вони за характеристиками дуже схожі на гравців, що платять. Мабуть, це той випадок, коли людина принципово донатити не буде. Або в іншому випадку, так як ці гравці найбільше

схожі на платячих гравців, то вони потребують нестандартних налаштувань, наприклад як полегшення гри шляхом зменшення цін на ігрові ресурси.

2.3.2 Висновок щодо кластеризації

В результаті є модель кластеризації, яка може назначити один з трьох кластерів неплатячим гравцям. Далі був здійснений деплой цієї моделі на сервер, та вона використовується самою грою в режимі реального часу.

Кожному кластеру були дані наступні імена для більш зручного користування:

0 – «0_slow». Група, яка дуже повільно грає. Щоб ускладнити гру є сенс збільшити кількість предметів.

1 – «0_skilled». Група для якої все легко. Щоб ускладнити гру є сенс зменшити час на рівні.

2 – «0_weak». Група, яка погано грає. Ця група за характеристиками більше схожа на людей, що платять. Можна припустити, сам тип гри їм подобається, але вони відчувають більше негативу, ніж позитиву через те, що їм складно. Тому можливо треба трохи полегшити їм гру шляхом зменшення цін на бустери, щоб вони мали трохи більше позитивних емоцій і вони більше витрачали. Робимо дешевше бустери шляхом зменшення шкали multi_use_mult_mod, яка збільшує ціну за кожен раз використання та змінимо шкалу level_mult, яка буде працювати на дистанції. діапазон залишиться тим самим, але буде менше дроблення.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНО-АНАЛІТИЧНИЙ

3.1 Зміна економіки

В табл. 2 наведені старі налаштування гри.

Параметр	Старе значення
Час	[80, 90, 70, 60, 65, 80, 75, 90, 85, 60]
Кількість предметів	[13, 15, 16, 17, 20, 18, 22, 19, 21, 19, 22, 18, 14, 20, 13, 18]
Бустер додатковий час	[0, 0.4, 0.8, 1.2, 1.6]
Бустер підсказка	[0, 0.2, 0.4, 0.6, 0.8, 1]
Бустер бомба	[0, 0.4, 0.8, 1.2, 1.6]
Бустер заморожування	[0, 0.7, 1.4, 2.1, 2.8]
Бустер ліхтар	[0, 0.7, 1.4, 2.1, 2.8]
Штраф за міскліки	[0, 0.4, 0.8, 1.2, 1.6]

Табл. 2 – старі налаштування гри

В табл. 3 наведені нові налаштування гри для кластера «0_skilled». Для цього кластера був зменшений час.

Параметр	Старе значення
Час	[50, 55, 40, 35, 40, 50, 45, 55, 50, 35]

Табл. 3 – нові налаштування для кластера «0_skilled»

В табл. 4 наведені нові налаштування гри для кластера «0_ slow». Для цього кластера було збільшено кількість предметів.

Параметр	Старе значення
Кількість предметів	[16, 19, 20, 21, 22, 21, 22, 21, 22, 20, 22, 20, 18, 22, 16, 21]

Табл. 4 – нові налаштування для кластера «0_slow»

В табл. 5 наведені нові налаштування гри для кластера «0_ weak». Для цього кластера були зменшені ціни на бустери та була зменшена ціна на штраф за міскліки.

Параметр	Старе значення
Бустер додатковий час	[0, 0.2, 0.45, 0.7, 0.95]
Бустер підсказка	[0, 0.1, 0.2, 0.35, 0.45, 0.6]
Бустер бомба	[0, 0.2, 0.45, 0.7, 0.95]
Бустер заморожування	[0, 0.4, 0.8, 1.2, 1.6]
Бустер ліхтар	[0, 0.4, 0.8, 1.2, 1.6]
Штраф за міскліки	[0, 0.2, 0.45, 0.7, 0.95]

Табл. 5 – нові налаштування для кластера «0_weak»

Нові значення були зменшені/збільшені за числом, що є співвідношенням booster_used у платників і конкретного кластера у тих, хто не платить. У наступних трьох строках можна побачити як ці значення розраховувались.

0_slow: $1 \div 1.33 = 0.8$

0_skilled: $0.66 \div 1.33 = 0.5$

0_weak: $0.87 \div 1.33 = 0.6$

3.2 Відомості щодо технічної реалізації

На рис. 47 зображена інформація щодо змінної first_level_payment. На основі цієї інформації був обраний поріг у вигляді 70 рівня, на якому буде відбуватися каліброва гравців. Було обрано саме це число, щоб якнайменш зламати баланс гри гравцям, для яких налаштування гри вже є оптимальними. Далі на 70 рівні якщо гравець неплатячий йому з однаковою вірогідністю назначається група a або b. У випадку групи a, для гравця гра не змінюється і він має кластер default. У випадку групи b гравцю назначається один з трьох

кластерів 0_skilled, 0_weak, 0_slow. І вже в залежності від кластеру гравцю імплементуються нові налаштування.

```
mean      51.539216
std       69.361683
min       10.000000
25%       14.000000
50%       26.500000
75%       58.750000
85%       76.550000
90%      117.000000
95%      188.900000
max      434.000000
Name: first_payment_level, dtype: float64
```

Рис. 47 – інформація щодо змінної first_level_payment

Визначення кластеру відбувається за допомогою вже відомої моделі за алгоритмом K-means. Вона була розміщена на сервері. Основний сервер гри зроблений на мові php. Тому спочатку йде запит з клієнтської частини до php серверу, далі за id гравця робиться вибірка історичних даних і передаються до python скрипту, де і розміщена модель. Далі відбувається визначення кластера, який заноситься в базу та повертається на клієнт.

3.3 A/B – тестування та результати експерименту

- Нульова гіпотеза – H_0 : p1 = p2 (кількість покупок при першому налаштуванні така сама, як і при другому).
- Альтернативна гіпотеза – H_1 : p1 ≠ p2 (кількість покупок при першому відрізняється від кількості покупок при другому налаштуванні).
- Рівень значимості – α: 0.05.

Рівень значимості був обраний стандартним для економічних розрахунків.

На основі отриманих значень, наведених у таблиці 6, потрібно розрахувати p-value та порівняти із рівнем значимості. Якщо $p < \alpha$, то є підстави відхилити

нульову гіпотезу на користь альтернативної. В оберненому випадку підстав для того, щоб відхилити нульову гіпотезу немає, тому що є досить велика вірогідність того, що отримана різниця результатів була отримана випадково. По ній можна побачити, що в групі **a** всього 49184 гравців та було здійснено 37 покупок, а в групі **b** всього 49272 гравців та було здійснено 52 покупки.

	Testing Group	
	a	**b**
Group amount	49184	49272
Purchase	37	52

Табл. 6 – отримані значення внаслідок A/B – тестування

На рис. 47 показано як було розраховано p-value за допомогою пакета statsmodels та метода proportions_ztest. p-value = 0.11.

```
[14] s1 = 37       #  кількість успіхів     вибірка A
     n1 = 49184      #  кількість випробувань   вибірка A
     s2 = 52        #  кількість успіхів     вибірка Б
     n2 = 49272      #  кількість випробувань   вибірка Б

     z1, p_value1 = sm.stats.proportions_ztest([s1, s2], [n1, n2])

     # p-значення
     print(p_value1)

     0.11358551026982616
```

Рис. 48 – розрахунок p-value

Так як, p-value $>\alpha$, то підстав для того, щоб відхилити нульову гіпотезу на користь альтернативної немає.

В таблицях 7 – 14 показані розширені економічні показники для всіх кластерів.

В таблиці 15 показані усереднені та підсумовані значення для всіх кластерів.

Money add (default):

Характеристика	Значення
level_complete	93.504.080
rating_prize	66.578.450
time_bonus	17.343.545
achievement_reward	8.833.500
chest_open	8.443.875
mini_tasks_reward	7.102.750
day_bonus	6.879.700
star_chest	6.017.950
video_reward	2.008.500
purchase	613.920
total	217.326.270

Average per user: 4231

Total users default: 44511

Табл. 7 – монет зароблено кластером default

Money remove (default):

Характеристика	Значення
buy_hint	62.864.850
buy_energy	31.425.800
buy_bomb	28.345.900
booster_buy	19.978.815
pay_penalty	19.580.035
buy_extra_time	14.948.550
buy_freeze	10.809.600
buy_chest_buy	6.333.225
buy_torch	1.606.640

total	195.893.415

Average per user: 4401

Total users default: 44511

Табл. 8 – монет витрачено кластером default

Money add (0_skilled):

Характеристика	Значення
level_complete	24.674.768
rating_prize	16.561.400
time_bonus	6.746.840
achievement_reward	2.481.500
chest_open	3.096.750
mini_tasks_reward	2.347.700
day_bonus	1.945.600
star_chest	1.358.600
video_reward	509.100
purchase	20.000
total	59.742.258

Average per user: 3908

Total users 0_skilled: 11197

Табл. 9 – монет зароблено кластером 0_skilled

Money remove (0_skilled):

Характеристика	Значення
buy_hint	10.478.250
buy_energy	7.145.400
buy_bomb	9.497.400
booster_buy	7.270.715
pay_penalty	2.147.875
buy_extra_time	11.187.000
buy_freeze	14.150.400

buy_chest_buy	2.498.475
buy_torch	606.900
total	64.982.415

Total users 0_skilled: 11197

Average per user: 5804

Табл. 10 – монет витрачено кластером 0_skilled

Money add (0_slow):

Характеристика	Значення
level_complete	1.112.832
rating_prize	1.333.700
time_bonus	395.540
achievement_reward	132.500
chest_open	269.250
mini_tasks_reward	175.300
day_bonus	185.000
star_chest	88.950
video_reward	93.300
purchase	20.000
total	3.786.372

Average per user: 2947

Total users 0_slow: 1285

Табл. 11 – монет зароблено кластером 0_slow

Money remove (0_slow):

Характеристика	Значення
buy_hint	668.325
buy_energy	138.200
buy_bomb	732.500
booster_buy	541.110
pay_penalty	250.725

buy_extra_time	537.550
buy_freeze	789.200
buy_chest_buy	143.250
buy_torch	606.900
total	3.800.860

Average per user: 4394

Total users 0_slow: 1285

Табл. 12 – монет витрачено кластером 0_slow

Money add (0_weak):

Характеристика	Значення
level_complete	41.101.232
rating_prize	33.940.400
time_bonus	8.849.260
achievement_reward	3.446.500
chest_open	4.358.375
mini_tasks_reward	3.807.700
day_bonus	4.342.400
star_chest	2.721.650
video_reward	1.188.600
purchase	90.000
total	103.846.117

Average per user: 3330

Total users 0_weak: 28509

Табл. 13 – монет зароблено кластером 0_weak

Money remove (0_weak):

Характеристика	Значення
buy_hint	32.710.990
buy_energy	12.823.200
buy_bomb	13.591.300

booster_buy	10.745.195
pay_penalty	13.913.603
buy_extra_time	8.392.875
buy_freeze	2.374.200
buy_chest_buy	2.100.150
buy_torch	461.860
total	97.113.373

Average per user: 3406

Total users 0_weak: 28509

Табл. 14 – монет витрачено кластером 0_weak

	default	0_weak	0_skilled	0_slow
Монет зароблено	4882	3642	5335	2947
Монет витрачено	4401	3406	5804	2957

Табл. 15 – усереднені та підсумовані показники заробітку для всіх кластерів

За цими значеннями можна побачити, що гравці кластеру default заробляють трішки більше, ніж витрачають. Гравці 0_weak також заробляють трішки більше, ніж витрачають. Гравці 0_skilled витрачають більше, ніж заробляють. Гравці 0_slow витрачають приблизно стільки ж, скільки заробляють.

3.4 Висновок щодо тестування

При проведенні A/B – тестування по факту покупок в групі b більше ніж в групі a. Але при проведенні z тесту пропорцій немає підстав відхилити нульову гіпотезу, яка тлумачить про те, що різниці між старими та новими налаштуваннями немає. При цьому видно, що змінилися економічні показники гравців, які грають за новими налаштуваннями, порівняно зі старими. Кластери 0_skilled та 0_slow почали більше витрачати монет і менше заробляти. Кластер 0_weak більше схожий на default, бо гра для них трішки була облегшена, на відміну від інших.

Тобто, нові налаштування все-таки впливають на гру гравців, а саме на економічні показники, в чому і був початковий задум. Гравці без ускладнення гри найчастіше витрачають монети на бустер hint, а у гравців, яким скоротили

час та збільшили кількість предметів, виріс сильно попит на бустер freeze. В цьому є сенс, бо freeze заморожує час до кінця рівня і дозволяє комфортно грати. Тому можливо, є сенс для проведення нового тестування в майбутньому з ще більш коригованими налаштуваннями. При використанні бустера freeze по суті відпадає сенс використовувати інші бустери, бо можна спокійно проходити рівень неквапливо шукаючи предмети. Можливо тоді, на дистанції гравці будуть в середньому менше приходити і у них буде більше мотивації здійснити першу покупку.

ВИСНОВОК

При спробі вирішити задачу класифікації були використані три алгоритми Random Forest, Logistic Regression та Support Vector Machines. Найліпше проявив себе алгоритм Random Forest, який має AUC score на тестових даних 0.8. Також варто зауважити, що модель вважає ключовими параметрами money_ratio, booster_used та win_rate, що також було підтверджено при розвідувальному аналізі даних. Результат навчання моделі можна розглядати як підтвердження того, що різниця між двома класами дійсно присутня.

При проведенні кластерного аналізу був застосований метод K-means. Вдалося розбити неплатячих гравців на три кластери:

- «0_slow». Група, яка дуже повільно грає. При цьому трошки більше заробляють ніж витрачають. При всіх показниках на дистанції непогано грають.
- «0_skilled». Група для якої все легко. Вони в середньому проходять багато рівнів, мало витрачають бустерів, багато заробляють монет та мало витрачають їх.
- «0_weak». Група, яка погано грає по всіх показниках. За своїми показниками дуже схожі на гравців платячих на момент їхньої першої покупки.

Після імплементації кластерної моделі було проведено A/B – тестування із такими вхідними даними:

- Нульова гіпотеза – H_0 : p1 = p2 (кількість покупок при першому налаштуванні така сама, як і при другому).
- Альтернативна гіпотеза – H_1 : p1 ≠ p2 (кількість покупок при першому відрізняється від кількості покупок при другому налаштуванні).
- Рівень значимості – α: 0.05.

Рівень значимості був обраний стандартним для економічних розрахунків.

Отримані результати тестування в таблиці 16.

	Testing Group	
	a	**b**
Group amount	49184	49272
Purchase	37	52

Табл. 16 – результати експерименту

p-value було отримано внаслідок проведення тесту пропорцій і дорівнює 0.11. Так як, p-value $>\alpha$, то підстав для того, щоб відхилити нульову гіпотезу на користь альтернативної немає.

У таблиці 17 зображені скорочені та усереднені економічні показники для різних кластерів.

	default	0_weak	0_skilled	0_slow
Монет зароблено	4882	3642	5335	2947
Монет витрачено	4401	3406	5804	2957

Табл. 17 – усереднені та підсумовані показники заробітку для всіх кластерів

За цими значеннями можна побачити, що гравці кластеру default заробляють трішки більше, ніж витрачають. Гравці 0_weak також заробляють трішки більше, ніж витрачають. Гравці 0_skilled витрачають більше, ніж заробляють. Гравці 0_slow витрачають приблизно стільки ж, скільки заробляють.

При проведенні A/B – тестування по факту покупок в групі b більше ніж в групі a. Але при проведенні z тесту пропорцій немає підстав відхилити нульову гіпотезу, яка тлумачить про те, що різниці між старими та новими налаштуваннями немає. При цьому видно, що змінилися економічні показники гравців, які грають за новими налаштуваннями, порівняно зі старими. Кластери 0_skilled та 0_slow почали більше витрачати монет і менше заробляти. Кластер 0_weak більше схожий на default, бо гра для них трішки була облегшена, на відміну від інших.

Тобто, нові налаштування все-таки впливають на гру гравців, а саме на економічні показники, в чому і був початковий задум. Гравці без ускладнення

гри найчастіше витрачають монети на бустер hint, а у гравців, яким скоротили час та збільшили кількість предметів, виріс сильно попит на бустер freeze. В цьому є сенс, бо freeze заморожує час до кінця рівня і дозволяє комфортно грати. Тому можливо, є сенс для проведення нового тестування в майбутньому з ще більш коригованими налаштуваннями. При використанні бустера freeze по суті відпадає сенс використовувати інші бустери, бо можна спокійно проходити рівень неквапливо шукаючи предмети. Можливо тоді, на дистанції гравці будуть в середньому менше приходити і у них буде більше мотивації здійснити першу покупку.

Також в результаті проведеного аналізу були підтверджені друга, четверта та п’ята гіпотези:

Win-rate.

Win-rate - це доля виграшів до всіх стартів рівнів. Було зроблено припущення, що гравці, які робили хоча б одну покупку, мають менший win-rate перед своєю першою покупкою, ніж ті, хто не платить.

Середня кількість монет, які гравець витрачає або заробляє за один пройдений рівень.

Було зроблено припущення, що гравці які платять, витрачають ігрових ресурсів більше ніж заробляють перед своєю першою покупкою, ніж ті, хто не платить.

Кількість бонусів, які гравець витрачає на рівні.

Було зроблено припущення, що гравці які платять, витрачають більше бонусів перед своєю першою покупкою, ніж ті, хто не платить. Це відбувається через те, що гра для таких гравців більш складна.

Цікавим моментам для інших розробників ігор жанру Hidden Objects може послугувати інформація на рис. 49.

	booster_used			money_ratio			win_rate		
	count	mean	median	count	mean	median	count	mean	median
is_paying									
0	1023	0.971338	0.789474	1023	1.470912	1.069035	1023	0.622926	0.641509
1	447	2.728507	1.333333	447	1.050710	0.745914	447	0.531015	0.500000

	level_time			day_passed		
	count	mean	median	count	mean	median
is_paying						
0	1023	69.006354	67.0	1023	6.825024	5.5
1	447	74.200224	66.0	447	8.866890	5.0

Рис. 49 – загальна зведена таблиця за таргет-класом та характеристиками

З цієї зведеної таблиці можна побачити, що за медіаною характеристики day_passed гравці обох класів в середньому проходять 5 рівнів в день. Тільки роблять це класи із різними зусиллями. Один клас особливо не напружується, а другий витрачає багато ігрових ресурсів та спроб, щоб досягти цієї мети. Тому я можу вважати, що для підвищення платежів в подібних іграх треба гравцям неплатячого класу намагатися зменшити число day_passed шляхом ускладнення гри. Тоді, маючи мотивацію проходити по 5 рівнів в день вони будуть витрачати більше ресурсів і можливо будуть більш відкриті до першої покупки.

На мою думку, є сенс в проведенні ще одного A/B – тестування. Видно, що нові налаштування вплинули на економіку гравців, але вони потребують ще коректив, бо і з цієї ситуації гравці знайшли вдалий вихід у вигляді бустера freeze. При великому попиті бустер повинен коштувати дорожче. Я вважаю, що для кластерів 0_skilled та 0_slow треба збільшити ціну на бустер freeze, бо через цей бустер нові налаштування вплинули не в повній мірі. Для кластеру 0_weak нема сенсу ускладнювати сильно гру, бо їм і так складно. Всі зміни для них повинні стосуватися виключно отриманих більш позитивних емоцій. Це можна зробити шляхом зміни часу гри на рівнях [80, 90, 70, 60, 65, 80, 75, 90, 85, 60]. Всього в сумі в цьому масиві 755 секунд. Треба зробити так, щоб ця сума

залишилася такою ж, але сам масив був більш волатильним. Тобто, щоб рівні чергувалися складними та легкими. Щоб у цих гравців в якийсь момент було більше позитивних емоції та впевненості в собі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Pandas. https://pandas.pydata.org/
2. Документація Numpy. https://numpy.org/
3. Документація Matplotlib. https://matplotlib.org/
4. Документація Python. https://docs.python.org/3/
5. Документація PIP. https://pypi.org/project/pip/
6. Форум для пошуку рішень. https://stackoverflow.com/
7. Документація Scikit Learn. https://scikit-learn.org/stable/
8. Документація Seaborn. https://seaborn.pydata.org/
9. Документація Statsmodels. https://www.statsmodels.org/stable/index.html
10. Розуміння AUC – ROC кривої. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c510.
11. Крива робочих характеристик приймача в оцінці діагностичних тестів. https://www.sciencedirect.com/science/article/pii/S1556086415306043#:~:text=AUC%20can%20be%20computed%20using%20the%20trapezoidal%20rule.&text=In%20general%2C%20an%20AUC%20of,than%200.9%20is%20considered%20outstanding.
12. What is Statistical Significance? https://byjus.com/maths/level-of-significance/#:~:text=Level%20of%20Significance%20Definition,with%20the%20outcomes%20of%20error.
13. What is the Level of Significance? https://www.youtube.com/watch?app=desktop&v=7KcnsAbcgnQ
14. Statistical Significance in A/B Testing – a Complete Guide. https://blog.analytics-toolkit.com/2017/statistical-significance-ab-testing-complete-guide/

The Null and the Alternative Hypotheses. https://www.tcc.fl.edu/media/divisions/learning-commons/resources-by-subject/math/statistics/The-Null-and-the-Alternative-Hypotheses.pdf

16. A/B-тестування: що це таке та чому вам варто його використовувати. https://hostiq.ua/blog/ukr/ab-testing/
17. How to Choose the Level of Significance: A Pedagogical Note.

https://mpra.ub.uni-muenchen.de/66373/1/MPRA_paper_66373.pdf

18. Розгортання моделі машинного навчання з використанням PHP та Python на веб-сторінці | Пратикша Джайн https://www.youtube.com/watch?v=anxZFB1k5zI&t
19. Мармоза А. Т. Теорія статистики підручник / А. Т. Мармоза – 2-ге вид. перероб. та доп. – К.: «Центр учбової літератури», 2022. – 592 с.
20. Ткач Є.І. Загальна теорія статистики: Підручник. – Тернопіль.: Лідер, 2004. – 388 с.
21. Лугінін О.Є. Статистика: Підручник. – К.: Центр учбової літератури, 2007. – 608 с.
22. Wes McKinney. Python for Data Analysis. Cambridge, 2012. – 470 p.
23. Peter Bruce, Andrew Bruce, and Peter Gedeck. Practical Statistics for Data Scientists Second Edition. – Sebastopol: O'Reilly Media, 2020. – 363 p.
24. Tukey, John W. The Collected Works of John W. Tukey. Vol. 4, Philosophy and Principles of Data Analysis: 1965–1986, edited by Lyle V. Jones. Boca Raton, Fla.: Chapman & Hall/CRC Press, 1987. – 650 p.
25. Waskom, Michael. «Seaborn: Statistical Data Visualization.» 2015.
26. Magy Seif El-Nasr; Truong-Huy D. Nguyen; Alessandro Canossa; Anders Drachen. Game Data Science. – Oxford: Oxford University Press, 2021. – 414 p

Додаток А. Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Найменування	Кількість аркушів	Примітки
1				
2		Документація	88	
3				
4	.КР.23.10.ПЗ	Пояснювальна записка	1	Формат А4
5				
6		Демонстраційний матеріал		Презентація на CD-R
7				
8		Копія роботи	1	Диск CD-R
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

					САУ.КР.YY.ZZ.ДА.ПЗ.
Змін.	Аркуш	№ докум.	Підпис	Дата	

Розроб.				**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Аркушів
К. розд.							
Керівн.					НТУ «ДП», 12; 124м-21		
Н.контр.							
Зав. каф.							

Запис **САУ.КР.YY.ZZ.ПЗ** означає наступне:

САУ – код випускаючої кафедри;

КР – кваліфікаційна робота;

N1 – загальна кількість сторінок пояснювальної записки кваліфікаційної роботи з додатками;

N2 – кількість аркушів демонстраційного матеріалу (слайдів презентації);

YY – рік захисту кваліфікаційної роботи в ЕК (наприклад “22”);

ZZ – номер теми студента в наказі про затвердження теми кваліфікаційної роботи (наприклад “06”);

ПЗ – пояснювальна записка;

ДА – додаток А;

12 – код галузі «Інформаційні технології».

Додаток Б

Відгук

на кваліфікаційну роботу магістра
студента(ки) групи *124м – 22 – 1*
спеціальності *124 Системний аналіз*

Тема кваліфікаційної роботи: _____

_____

Обсяг кваліфікаційної роботи _____стор.

Мета кваліфікаційної роботи: _____

_____

Актуальність теми _____

_____

_____

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 124 Системний аналіз, оскільки _____

_____

_____

Виконані в кваліфікаційній роботі завдання відповідають вимогам освітньо-кваліфікаційного рівня магістра. Оригінальність наукових рішень полягає в _____

_____

_____

_____

Практичне значення результатів кваліфікаційної роботи полягає в _____

_____

_____

Висновки підтверджують можливість використання результатів роботи в _____

_____

_____

Оформлення пояснювальної записки та демонстраційного матеріалу до неї виконано згідно з вимогами. Роботу виконано самостійно, відповідно до завдання та у повному обсязі (*в разі невідповідності – вказати*)

У роботі відзначено такі недоліки: _____

_____

_____

Кваліфікаційна робота в цілому заслуговує оцінки:_____

З урахуванням висловлених зауважень автор (не) заслуговує присвоєння кваліфікації «магістр з системного аналізу».

Керівник кваліфікаційної роботи магістра,

науковий ступінь, вчене звання, посада _____/ ПІБ

Додаток В

Рецензія

на кваліфікаційну роботу магістра
студента групи *124м – 22 – 1* спеціальності
124 Системний аналіз

Тема кваліфікаційної роботи:

_____
_____
_____

Обсяг кваліфікаційної роботи: _____

Висновок про відповідність кваліфікаційної роботи завданню та освітньо-професійній програмі спеціальності _____

_____
_____

Загальна характеристика кваліфікаційної роботи, ступінь використання нормативно-методичної літератури та передового досвіду

_____
_____
_____

Позитивні сторони кваліфікаційної роботи:

_____
_____
_____

Основні недоліки кваліфікаційної роботи:

_____
_____
_____
_____

Кваліфікаційна робота в цілому заслуговує оцінки: _____

З урахуванням висловлених зауважень автор (не) заслуговує присвоєння кваліфікації «магістр з системного аналізу».

Рецензент,
науковий ступінь, вчене звання, посада _____/ ПІБ

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
y_train.value_counts(normalize=True)
n_estimators=[100, 150, 200]
max_features= [2, 3]
bootstrap = [True,False]
oob_score = [True,False]
min_samples_split = [5, 6, 7]
min_samples_leaf = [5, 6, 7]
max_depth = [5, 6]
class_weight = [{0:0.3,1:0.7}]
param_grid = {
    'n_estimators':n_estimators,
    'max_features':max_features,
    'bootstrap':bootstrap,
    'oob_score':oob_score,
    'max_depth':max_depth,
    'min_samples_split':min_samples_split,
    'min_samples_leaf':min_samples_leaf,
    "class_weight":class_weight
}
model = RandomForestClassifier()
grid = GridSearchCV(model,param_grid)
grid.fit(X_train,y_train)
grid.best_params_
# train_pred = grid.predict(X_train)
threshold = 0.4

train_predicted_proba = grid.predict_proba(X_train)
train_predictions_exp = (train_predicted_proba [:,1] >= threshold).astype('int')
from sklearn.metrics import
```

```
confusion_matrix,classification_report,ConfusionMatrixDisplay,accuracy_score,
roc_curve, RocCurveDisplay, auc
print(classification_report(y_train,train_predictions_exp))
cm = confusion_matrix(y_train, train_predictions_exp, labels=grid.classes_)
disp = ConfusionMatrixDisplay(cm,display_labels=grid.classes_)
disp.plot()
fpr, tpr, thresholds = roc_curve(y_train,train_predictions_exp)
roc_auc = auc(fpr, tpr)
display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,
estimator_name='Random forest')
display.plot()
feature_imp = pd.DataFrame(grid.best_estimator_.feature_importances_, index =
X_train.columns, columns = ['importance'])
feature_imp.sort_values(by = 'importance', ascending = False)
# test_pred = grid.predict(X_test)
threshold = 0.4

test_predicted_proba = grid.predict_proba(X_test)
test_predictions_exp = (test_predicted_proba [:,1] >= threshold).astype('int')
print(classification_report(y_test,test_predictions_exp))
fpr, tpr, thresholds = roc_curve(y_test,test_predictions_exp)
roc_auc = auc(fpr, tpr)
display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,
estimator_name='Random forest')
display.plot()
cm = confusion_matrix(y_test, test_predictions_exp, labels=grid.classes_)
disp = ConfusionMatrixDisplay(cm,display_labels=grid.classes_)
disp.plot()
```

Додаток Д. Код кластеризації

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
from statsmodels.stats.outliers_influence import variance_inflation_factor

df = pd.read_csv('../df_scaled.csv')
df.head()

df = df.set_index('id')

df = df[df['is_paying'] == 0]

df_new = df.drop('is_paying', axis = 1)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df_new_scaled = pd.DataFrame(scaler.fit_transform(df_new), columns=df_new.columns)

from sklearn.cluster import KMeans

ssd = []

for k in range(2,11):

    model = KMeans(n_clusters=k, random_state=42)
```

```
    model.fit(df_new_scaled)

    #Sum of squared distances of samples to their closest cluster center.
    ssd.append(model.inertia_)

plt.figure()
plt.plot(range(2,11),ssd,'o--')
plt.xlabel("K Value")
plt.ylabel("Sum of Squared Distances")

pd.Series(ssd).diff()

model = KMeans(n_clusters=3, random_state=42)
clusters = model.fit_predict(df_new_scaled)
len(clusters)

import joblib

df['cluster'] = clusters

cm = sns.diverging_palette(10, 133, as_cmap=True)
df.groupby('cluster')[df.columns[:-
1]].median().transpose().style.background_gradient(cmap=cm, axis = 1)

if __name__ == "__main__":
    scaler = MinMaxScaler()
    df_new_scaled                =                pd.DataFrame(scaler.fit_transform(df_new),
columns=df_new.columns)
```

```
model = KMeans(n_clusters=3, random_state=42)
clusters = model.fit_predict(df_new_scaled)
joblib.dump(model, 'model.pkl')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(list(df_new_scaled.columns), 'col_names.pkl')
```

```
import numpy as np
import pandas as pd
import joblib
import ast
import sys
import pathlib
import warnings
#scikit-learn
warnings.filterwarnings('ignore')

full_path = pathlib.Path(__file__).parent.resolve()

global_info = {
    "error": False
}
cols = ['id',
'achievement_reward_collect',
'avg_day_levels_passed',
'avg_level_time',
'avg_win_rate',
'bomb_used',
'booster_buy',
'booster_use',
'browser_name',
'buy_energy',
'chest_open_collect',
'current_riches',
'day_bonus_collect',
'day_levels_passed_new',
```

'day_levels_passed',
'extra_time_used',
'first_name',
'first_payment_level',
'freeze_used',
'game_id',
'gender',
'hint_used',
'level_complete_collect',
'level_failed',
'level_failed_missclick',
'level_failed_quit',
'level_failed_time',
'level_time',
'locale',
'lost',
'mini_tasks_reward_collect',
'money_add',
'money_remove',
'new_user_entry_point',
'pay_penalty',
'pay_penalty_final',
'pay_penalty_normal',
'pay_penalty_total',
'penalty_factor',
'rating_prize_collect',
'star_chest_collect',
'start',
'start_day',
'time_bonus_collect',
'torch_used',
'total_levels_24h',

```
'total_payments',
'video_reward_collect',
'win',
'win_rate']

cols_to_scale = [
    'achievement_reward_collect', 'booster_buy', 'buy_energy', 'chest_open_collect',
    'day_bonus_collect',
    'level_complete_collect',
    'mini_tasks_reward_collect', 'pay_penalty', 'rating_prize_collect', 'bomb_used',
'extra_time_used',
    'star_chest_collect', 'time_bonus_collect', 'video_reward_collect', 'freeze_used',
'hint_used', 'torch_used'
]

def transform_data(data):
    df = pd.DataFrame(columns=cols)

    rows = []
    for i in data:
        row = { "id": i }
        for key in data[i].keys():
            row[key] = data[i][key]
        rows.append(row)

    df_new = pd.DataFrame(rows)
    df = pd.concat([df, df_new])

    df['game_id'] = df['game_id'].fillna('')
    df['start_day'] = df['start_day'].fillna(-1)
```

```
    df['locale'] = df['locale'].fillna('')
    df['new_user_entry_point'] = df['new_user_entry_point'].fillna('')
    df['current_riches'] = df['current_riches'].apply(lambda x: {} if pd.isnull(x) else x)
    df['browser_name'] = df['browser_name'].fillna('')
    df['level_failed'] = df['level_failed'].apply(lambda x: {} if pd.isnull(x) else x)
    df['level_time'] = df['level_time'].apply(lambda x: {} if pd.isnull(x) else x)
    df['day_levels_passed'] = df['day_levels_passed'].apply(lambda x: {} if pd.isnull(x) else x)
    df['win_rate'] = df['win_rate'].apply(lambda x: {} if pd.isnull(x) else x)
    df['money_add'] = df['money_add'].apply(lambda x: {} if pd.isnull(x) else x)
    df['money_remove'] = df['money_remove'].apply(lambda x: {} if pd.isnull(x) else x)
    df['booster_use'] = df['booster_use'].apply(lambda x: {} if pd.isnull(x) else x)
    df['penalty_factor'] = df['penalty_factor'].apply(lambda x: {} if pd.isnull(x) else x)
    df['total_levels_24h'] = df['total_levels_24h'].fillna(-1)
    df['first_name'] = df['first_name'].fillna('')
    df['total_payments'] = df['total_payments'].fillna(0)
    df['first_payment_level'] = df['first_payment_level'].fillna(-1)

    df['achievement_reward_collect'] = df['achievement_reward_collect'].fillna(0)
    df['avg_day_levels_passed'] = df['avg_day_levels_passed'].fillna(0)
    df['avg_level_time'] = df['avg_level_time'].fillna(0)
    df['avg_win_rate'] = df['avg_win_rate'].fillna(0)
    df['bomb_used'] = df['bomb_used'].fillna(0)
    df['booster_buy'] = df['booster_buy'].fillna(0)
    df['buy_energy'] = df['buy_energy'].fillna(0)
    df['chest_open_collect'] = df['chest_open_collect'].fillna(0)
    df['day_bonus_collect'] = df['day_bonus_collect'].fillna(0)
    df['day_levels_passed_new'] = df['day_levels_passed_new'].fillna(0)
    df['extra_time_used'] = df['extra_time_used'].fillna(0)
    df['freeze_used'] = df['freeze_used'].fillna(0)
    df['hint_used'] = df['hint_used'].fillna(0)
```

```
df['level_failed_missclick'] = df['level_failed_missclick'].fillna(0)
df['level_failed_quit'] = df['level_failed_quit'].fillna(0)
df['level_failed_time'] = df['level_failed_time'].fillna(0)
df['lost'] = df['lost'].fillna(0)
df['mini_tasks_reward_collect'] = df['mini_tasks_reward_collect'].fillna(0)
df['pay_penalty'] = df['pay_penalty'].fillna(0)
df['pay_penalty_final'] = df['pay_penalty_final'].fillna(0)
df['pay_penalty_normal'] = df['pay_penalty_normal'].fillna(0)
df['pay_penalty_total'] = df['pay_penalty_total'].fillna(0)
df['rating_prize_collect'] = df['rating_prize_collect'].fillna(0)
df['star_chest_collect'] = df['star_chest_collect'].fillna(0)
df['start'] = df['start'].fillna(0)
df['time_bonus_collect'] = df['time_bonus_collect'].fillna(0)
df['torch_used'] = df['torch_used'].fillna(0)
df['video_reward_collect'] = df['video_reward_collect'].fillna(0)
df['win'] = df['win'].fillna(0)

df = df.apply(extract_level_time_val, axis = 1)
df = df.apply(extract_day_levels_passed_val, axis = 1)
df = df.apply(extract_win_rate_val, axis = 1)
df = df.apply(extract_money_add_val, axis = 1)
df = df.apply(extract_money_remove_val, axis = 1)
df = df.apply(extract_penalty_factor_val, axis = 1)
df = df.apply(extract_booster_use_val, axis = 1)
df = df.apply(extract_level_failed_val, axis = 1)

df = df.drop(['level_failed', 'level_time', 'day_levels_passed', 'win_rate', 'penalty_factor', 'money_add', 'money_remove', 'booster_use', 'current_riches'], axis = 1)
df = df.drop(['browser_name', 'first_name', 'game_id', 'gender', 'locale', 'new_user_entry_point', 'start_day', 'lost', 'start', 'win'], axis = 1)
```

```
    df = df.set_index('id')
    df.rename(columns={'day_levels_passed_new': 'day_levels_passed'}, inplace=True)

    for col in cols_to_scale:
        df[col] = df[col] / df['day_levels_passed']

    df['booster_used'] = (df['bomb_used'] + df['hint_used'] + df['torch_used'] + df['freeze_used'] + df['extra_time_used'])


    df = df.drop(['bomb_used', 'hint_used', 'torch_used', 'freeze_used', 'extra_time_used'], axis = 1)
    df = df.drop(['day_levels_passed', 'first_payment_level'], axis = 1)
    df['money_collect'] = (df['day_bonus_collect'] + df['chest_open_collect'] + df['achievement_reward_collect'] + df['mini_tasks_reward_collect'] + df['star_chest_collect'] + df['time_bonus_collect'] + df['rating_prize_collect'] + df['video_reward_collect'] + df['level_complete_collect'])
    df = df.drop(['video_reward_collect', 'rating_prize_collect', 'time_bonus_collect', 'star_chest_collect', 'mini_tasks_reward_collect', 'achievement_reward_collect', 'chest_open_collect', 'day_bonus_collect', 'level_complete_collect'], axis = 1)
    df['money_remove'] = (df['booster_buy'] + df['buy_energy'] + df['pay_penalty'])
    df = df.drop(['booster_buy', 'buy_energy', 'pay_penalty'], axis = 1)
    df['money_ratio'] = df['money_collect'] / df['money_remove']
    df = df.drop(['money_collect', 'money_remove'], axis = 1)
    df.rename(columns={'avg_day_levels_passed': 'day_passed', 'avg_level_time': 'level_time', 'avg_win_rate': 'win_rate', 'total_levels_24h': 'first_day_passed'}, inplace=True)

    return df

def extract_level_time_val(df):
```

```
    level_time = df['level_time']
    keys = level_time.keys()
    values = []
    for key in keys:
        try:
            values.append(level_time[key])
        except TypeError:
            print('error')

    new_key = 'avg_level_time'
    df[new_key] = np.median(values)
    return df

def extract_day_levels_passed_val(df):
    day_levels_passed = df['day_levels_passed']
    keys = day_levels_passed.keys()
    total = 0
    values = []
    for key in keys:
        values.append(day_levels_passed[key])
        total = total + day_levels_passed[key]

    new_key = 'avg_day_levels_passed'
    df[new_key] = np.median(values)
    new_key = 'day_levels_passed_new'
    df[new_key] = total
    return df

def extract_win_rate_val(df):
    win_rate = df['win_rate']
```

```
    keys = win_rate.keys()
    start = 0
    win = 0
    lost = 0
    values = []
    for key in keys:
        if 'win' in win_rate[key]: win = win + win_rate[key]['win']
        if 'start' in win_rate[key]: start = start + win_rate[key]['start']
        if 'lost' in win_rate[key]: lost = lost + win_rate[key]['lost']

    new_key = 'avg_win_rate'
    try:
        avg_win_rate = win/start
    except ZeroDivisionError:
        avg_win_rate = -1
    df[new_key] = avg_win_rate

    df['win'] = win
    df['lost'] = lost
    df['start'] = start
    return df

def extract_money_add_val(df):
    money_add = df['money_add']
    obj = {}
    keys = money_add.keys()
    for key in keys:
        keys2 = money_add[key].keys()
        for key2 in keys2:
            if key2 not in obj: obj[key2] = 0
            obj[key2] = obj[key2] + money_add[key][key2]
```

```
    final_keys = obj.keys()
    for final_key in final_keys:
#         print(obj[final_key], final_key)
        new_key = final_key + '_collect'
        df[new_key] = obj[final_key]
    return df

def extract_money_remove_val(df):
    money_remove = df['money_remove']
    obj = {}
    keys = money_remove.keys()
    for key in keys:
        keys2 = money_remove[key].keys()
        for key2 in keys2:
            if key2 not in obj: obj[key2] = 0
            obj[key2] = obj[key2] + np.abs(money_remove[key][key2])

    final_keys = obj.keys()
    for final_key in final_keys:
        new_key = final_key
        df[new_key] = obj[final_key]
    return df

def extract_penalty_factor_val(df):
    penalty_factor = df['penalty_factor']
    obj = {}
    keys = penalty_factor.keys()
    for key in keys:

        keys2 = penalty_factor[key].keys()
```

```
        for key2 in keys2:
            if key2 not in obj: obj[key2] = 0
            obj[key2] = obj[key2] + np.abs(penalty_factor[key][key2])

    final_keys = obj.keys()
    for final_key in final_keys:
        new_key = 'pay_penalty_' + final_key
        df[new_key] = obj[final_key]
    return df

def extract_booster_use_val(df):
    booster_use = df['booster_use']
    obj = {}
    keys = booster_use.keys()
    for key in keys:
        keys2 = booster_use[key].keys()
        for key2 in keys2:
            if key2 not in obj: obj[key2] = 0
            obj[key2] = obj[key2] + np.abs(booster_use[key][key2])

    final_keys = obj.keys()
    for final_key in final_keys:
        new_key = final_key + '_used'
        df[new_key] = obj[final_key]
    return df

def extract_level_failed_val(df):
    level_failed = df['level_failed']
    obj = {}
    keys = level_failed.keys()
#     print(keys)
    for key in keys:
```

```
#         print(key, level_failed[key])
        new_key = 'level_failed_' + key
        df[new_key] = level_failed[key]

    return df

def get_cluster(val):
    if val == 0: return '0_slow'
    elif val == 1: return '0_skilled'
    elif val == 2: return '0_weak'
    else: return 'no_info'

def error_catch():
    global_info['error'] = True
    print("{\"success\": true, \"prediction\":" + "\"" + "0_skilled" + "\"}")

if __name__ == '__main__':
    try:
        if not global_info['error']:
            path = ''.join(sys.argv[1])
            with open(path) as my_file:
                # print(my_file.read())
                content = my_file.read()
    except Exception as ex:
        # print('{"success": false, "reason": "error with reading file"}')
        error_catch()
    try:
        if not global_info['error']:
            # preload model
            # col_names = joblib.load('col_names.pkl')
```

```
        col_names = joblib.load(f'{full_path}/col_names.pkl')
        # model = joblib.load('model.pkl')
        model = joblib.load(f'{full_path}/model.pkl')
        # scaler = joblib.load('scaler.pkl')
        scaler = joblib.load(f'{full_path}/scaler.pkl')
    except Exception as ex:
        error_catch()
        # print("{\"success\": false, \"reason\": \"Error with preloading model\", \"error\":"
+ "\"" + str(ex) + "\"}")
        # print('{"success": false, "reason": "wrong data"}')
    try:
        if not global_info['error']:
            # feat_data = request.json
            # print(sys.argv[1])
            # param_sliced = sys.argv[1:]
            # param = ".join(sys.argv[1])
            param = content
            # param = sys.argv[1]
            # param = param.replace("\\{", "{")
            # param = param.replace("\\}", "}")
            # param = param.replace("\\", "\"")

            param = param.replace("@", "\"")
            param = "{" + param + "}"
            # print(param)
            # print(".join(sys.argv[1:]))
            # print(sys.argv[1:])
            # convert string to python dict
            feat_data = ast.literal_eval(param)
            # print(feat_data)
            # print(sys.argv[1])
    except Exception as ex:
```

```
        error_catch()
        # print("{\"success\": false, \"reason\": \"Error with converting string to python dict\", \"error\":" + "\"" + str(ex) + "\"}")

    try:
        if not global_info['error']:
            # transformation
            df = transform_data(feat_data)
            # print("300")
    except Exception as ex:
        error_catch()
        # print("{\"success\": false, \"reason\": \"Error with transformation data\", \"error\":" + "\"" + str(ex) + "\"}")

    try:
        if not global_info['error']:
            # scaling
            df_scaled = pd.DataFrame(scaler.transform(df[col_names]), columns=col_names)
    except Exception as ex:
        error_catch()
        # print("{\"success\": false, \"reason\": \"Error with scaling\", \"error\":" + "\"" + str(ex) + "\"}")

    try:
        if not global_info['error']:
            # checking if all is allright
            if df_scaled.isnull().values.any() == True:
                print("{\"success\": false, \"reason\": \"wrong data\"}")
            else:
```

```
            # predicting
            prediction = list(model.predict(df_scaled))[0]
            prediction = get_cluster(prediction)
            print("{\"success\": true, \"prediction\":" + "\"" + str(prediction) + "\"}")
    except Exception as ex:
        error_catch()
        # print("{\"success\": false, \"reason\": \"Error with predicting\", \"error\":" + "\"" + str(ex) + "\"}")
```