НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Прогнозування рейтенгу настольної гри в залежності від параметрів»

Студента 2 курсу ІП-13 групи
Спеціальності: 121
«Інженерія програмного забезпечення»
Дейнега Владислав Миколайович

«ПРИЙНЯВ» з оцінкою

_____

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

_____
Підпис Дата

Київ - 2023 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-13 Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Дейнеги Владислава Миколайовича

1.Тема роботи Прогназування рейтенгу настольної гри в залежності від Параметрів.

2.Строк здачі студентом закінченої роботи 08.06.2022

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту https://www.kaggle.com/datasets/threnjen/board-games-database-from-boardgamegeek?select=games.csv

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1.Постановка задачі

2.Аналіз предметної області

3.Інтелектуальний аналіз даних

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

6.Дата видачі завдання 30.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	30.03.2023	
2.	Визначення зовнішніх джерел даних	30.03.2023	
3.	Пошук та вивчення літератури з питань курсової роботи	06.06.2023	
4.	Обробка та аналіз даних	06.06.2023	
5.	Обґрунтування методів інтелектуального аналізу даних	06.06.2023	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	06.06.2023	
7.	Підготовка пояснювальної записки	07.06.2023	
8.	Здача курсової роботи на перевірку	08.06.2023	
9.	Захист курсової роботи		

Студент _____ Дейнега В.Д.

(підпис) (прізвище, ім’я, по батькові)

Керівник _____ доц. Ліхоузова Т.А

(підпис) (прізвище, ім’я, по батькові)

Керівник _____ доц. Олійник Ю.О.

(підпис) (прізвище, ім’я, по батькові)

"9" червня 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 33 сторінок, 17 рисунки, 8 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз даних з подальшим прогнозуванням та графічним відображенням результатів.

Мета роботи: пошук, аналіз та обробка даних, реалізація ПЗ, що використовує отримані дані для подальшого аналізу та прогнозування результату.

Дана курсова робота включає в себе: постановку задачі, аналіз предметної області, роботу з даними, аналіз обраних методів для їх графічного відображення, прогнозування та їх порівняння.

ДАТАСЕТ, ПРОГНОЗУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, LINEARREGRESSION,DECISIONTREEREGRESSOR,RANDOMFORESTREGRESSOR.

ЗМІСТ

ВСТУП

Успішність настільних ігор займає особливе місце в середовищі геймдеву, привертаючи увагу як гравців, так і творців ігор. Відповідно, прогнозування рейтингу настольних ігор на основі їх параметрів стає ключовим етапом у процесі створення і розвитку таких ігор.

Настільні ігри володіють своїм унікальним чаром та можуть надати незабутні враження гравцям. Однак, їх успішність на ринку може залежати від різних факторів, таких як складність, рік випуску, категорії та інші. Таким чином, важливо мати засоби та методи для прогнозування рейтингу настольних ігор на етапі їх розробки.

У цій роботі будуть використані дані про настольні ігри, їх параметри та рейтинги, які будуть піддані аналізу та моделюванню. У результаті отримаємо моделі, які з деякою точністю зможуть прогнозувати успішність гри.

Дана курсова робота буде розроблена з використанням технологій та бібліотек Python 3[1], Pandas[2], Matplotlib[3], Sklearn[4].

1.ПОСТАНОВКА ЗАДАЧІ

Під час виконання курсової роботи необхідно виконати наступні завдання: аналізу предметної області, обробка датасета: завантаження, дослідження його структури; фільтрування даних, що не мають впливу на вихідний результат; вибір методів для прогнозування та не коректних даних; аналіз отриманих результатів кожного з методів та порівняння отриманих результатів ефективності.

Створення застосунку, що графічно відображає отримані дані, поділяє отримані дані на тренувальні та тестові та проводить їх аналіз для отримання передбачення за допомогою різних моделей прогнозування.

Для прогнозування були обрані методи Linear Regression[5], DecisionTreeRegressor[6], RandomForestRegressor[7]. Результати будуть оцінені середньоквадратичною помилкою (mean squared error) та коефіцієнтом детермінації (R2 score) і потім обрати найбільш оптимальний метод.

Результат дослідження можна використористовувати для прогнозування успішності настільної ігри, що буде корисним для видавців та творців настільних ігор.

Вхідний датасет має близька 40 колонок з данними, але використовувати для прогнозування має сенс використовувати рік випуску, складність гри, чи належить гра до одної із категорій: тематична, стратегія, військова, сімейна, карткові, абстрактна, party game, дитяча. Всі інші дані є не доцільним використовувати, оскільки вони мають малий вплив на рейтинг.

Вихідними даними будуть передбачення рейтенгу гри.

2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Прогнозування успішності настільних ігор є актуальною та значущою задачею для творців настільних ігор. З ростом популярності настільних ігор серед різних груп користувачів, таких як сімейні гравці, геймери або спеціалісти з галузі розвитку та навчання, зростає і конкуренція на ринку. У таких умовах творці ігор потребують ефективних інструментів для оцінки та прогнозування успішності своїх продуктів.

Дослідження в галузі прогнозування успішності настільних ігор вже отримало певний прогрес. Було проведено декілька досліджень, в яких використовувалися різні методи, такі як машинне навчання, статистичні моделі та аналіз даних, для прогнозування рейтингу та популярності ігор. Результати цих досліджень показали, що параметри гри, такі як час гри, кількість гравців, складність правил, тематика та інші фактори, можуть впливати на успішність ігри серед гравців.

Проте, існують деякі виклики та обмеження, з якими можуть зіткнутися дослідники та творці настільних ігор при прогнозуванні рейтингу. Наприклад, доступ до великого обсягу даних про ігри та їх параметри може бути обмеженим. Також, належне визначення та вимірювання поняття успішності чи популярності ігри може бути складним завданням.

З урахуванням вищезазначених факторів, в цій курсовій роботі ми спробуємо розв'язати задачу прогнозування рейтингу настільних ігор на основі їх параметрів.

3. РОБОТА З ДАНИМИ

3.1 Опис обраних даних

Для вирішення поставленої задачі прогнозування рейтенгу гри в залежності від параметрів був використаний датасет “ Board Game Database from BoardGameGeek “. Цей датасет має 48 колонок, але цікаві для нас є:

- BGGId – ID гри на сайті Board Games Geek
- Name – назва гри
- AvgRating – середній рейтинг гри
- YearPublished – рік публікації
- GameWeight – складність гри
- Cat_Thematic – належність гри до категорії «Тематичні»
- Cat_Strategy - належність гри до категорії «Стратегії»
- Cat_War - належність гри до категорії «Військові»
- Cat_Family - належність гри до категорії «Сімейні»
- Cat_CGS - належність гри до категорії «Карткові»
- Cat_Abstract - належність гри до категорії «Абстрактні»
- Cat_Party - належність гри до категорії «Party Game»

Ці колонки підходять для прогнозування рейтенгу настільної гри, оскільки вони мають інформацію про складність гри та належність до одної або декількох категорій і інформацію, яка допоможе очистити датасет від дублювання інформації. Вся інша інформація цього датасету не може допомогти у покращенню прогнозу або навпаки буде погіршувати результат.

3.2 Перевірка даних

Тепер завантажимо та оглянемо наші дані. Для цього скористаємось можливостями бібліотеки “Pandas”.

```
dataset = pd.read_csv('D:/Git/Kursova/data/games.csv')
dataset = dataset[['BGGId', 'Name', 'AvgRating', 'YearPublished', 'GameWeight', 'Cat_Thematic', 'Cat_Strategy', 'Cat_War',
            'Cat_Family', 'Cat_CGS', 'Cat_Abstract', 'Cat_Party']]

print("            Інформація")
print(dataset.info())
```

```
            Інформація
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21925 entries, 0 to 21924
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   BGGId          21925 non-null  int64
 1   Name           21925 non-null  object
 2   AvgRating      21925 non-null  float64
 3   YearPublished  21925 non-null  int64
 4   GameWeight     21925 non-null  float64
 5   Cat_Thematic   21925 non-null  int64
 6   Cat_Strategy   21925 non-null  int64
 7   Cat_War        21925 non-null  int64
 8   Cat_Family     21925 non-null  int64
 9   Cat_CGS        21925 non-null  int64
 10  Cat_Abstract   21925 non-null  int64
 11  Cat_Party      21925 non-null  int64
dtypes: float64(2), int64(9), object(1)
memory usage: 2.0+ MB
None
```

Рисунок 3.1 – загрузка датафрейму та інформація про нього

Ми можемо побачити, що дані не мають пустих значеннь та типи даних визначено вірно.

Далі подивимось інформацію про числові дані

```
print("    Опис")
print(dataset.describe())
```

```
    Опис
              BGGId     AvgRating  YearPublished    GameWeight  Cat_Thematic  \
count   21925.000000  21925.000000   21925.000000  21925.000000  21925.000000
mean   117652.663216      6.424922    1985.494914      1.982131      0.055827
std    104628.721777      0.932477     212.486214      0.848983      0.229592
min         1.000000      1.041330   -3500.000000      0.000000      0.000000
25%     12346.000000      5.836960    2001.000000      1.333300      0.000000
50%    105305.000000      6.453950    2011.000000      1.968800      0.000000
75%    206169.000000      7.052450    2017.000000      2.525200      0.000000
max    349161.000000      9.914290    2021.000000      5.000000      1.000000

       Cat_Strategy       Cat_War    Cat_Family       Cat_CGS  Cat_Abstract  \
count   21925.00000  21925.000000  21925.000000  21925.000000  21925.000000
mean        0.10577      0.161003      0.105633      0.013820      0.050855
std         0.30755      0.367542      0.307374      0.116745      0.219707
min         0.00000      0.000000      0.000000      0.000000      0.000000
25%         0.00000      0.000000      0.000000      0.000000      0.000000
50%         0.00000      0.000000      0.000000      0.000000      0.000000
75%         0.00000      0.000000      0.000000      0.000000      0.000000
max         1.00000      1.000000      1.000000      1.000000      1.000000

         Cat_Party
count  21925.000000
mean       0.029190
std        0.168344
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
```

Рисунок 3.2 - інформацію про числові дані

Також додатково виведемо інформацію про кількість настільних ігор по роках.

```
year_info = dataset['YearPublished'].value_counts().sort_index()
plt.plot(year_info.index, year_info.values, marker='o')
plt.xlabel('Роки')
plt.ylabel('Кількість ігор')
plt.title('Кількість ігор за рік')
plt.grid(True)
plt.show()
```

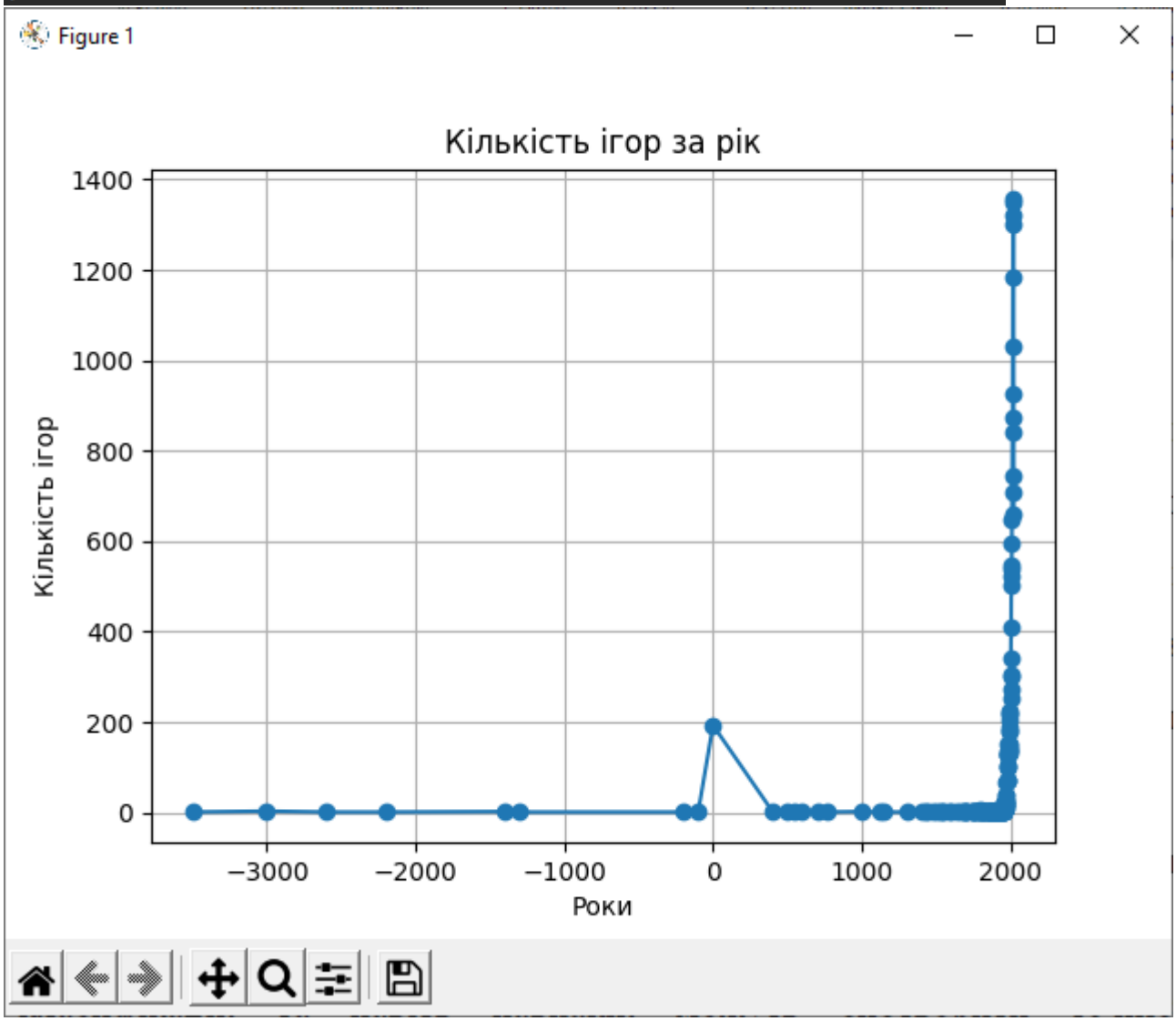

Рисунок 3.3 – кількість ігор випущених по роках

Після огляду інформації про датафрейм та його числові данні, ми бачимо, що дані про рейтинг є коректними, оскільки число варується від 0 до 10, що є цілком нормально для рейтенгу. Рік видавництва має від'ємні значення, що може бути роками до нашої ери, але у нашій задачі це буде тільки заважати. Тому ми відкинемо всі настільні ігри до 1744 року, оскільки найбільша частина ігор була випущена після цього року.

```
dataset = dataset[dataset['YearPublished'] >= 1900]

year_info = dataset['YearPublished'].value_counts().sort_index()
plt.plot(year_info.index, year_info.values, marker='o')
plt.xlabel('Роки')
plt.ylabel('Кількість ігор')
plt.title('Кількість ігор за рік')
plt.grid(True)
plt.show()
```

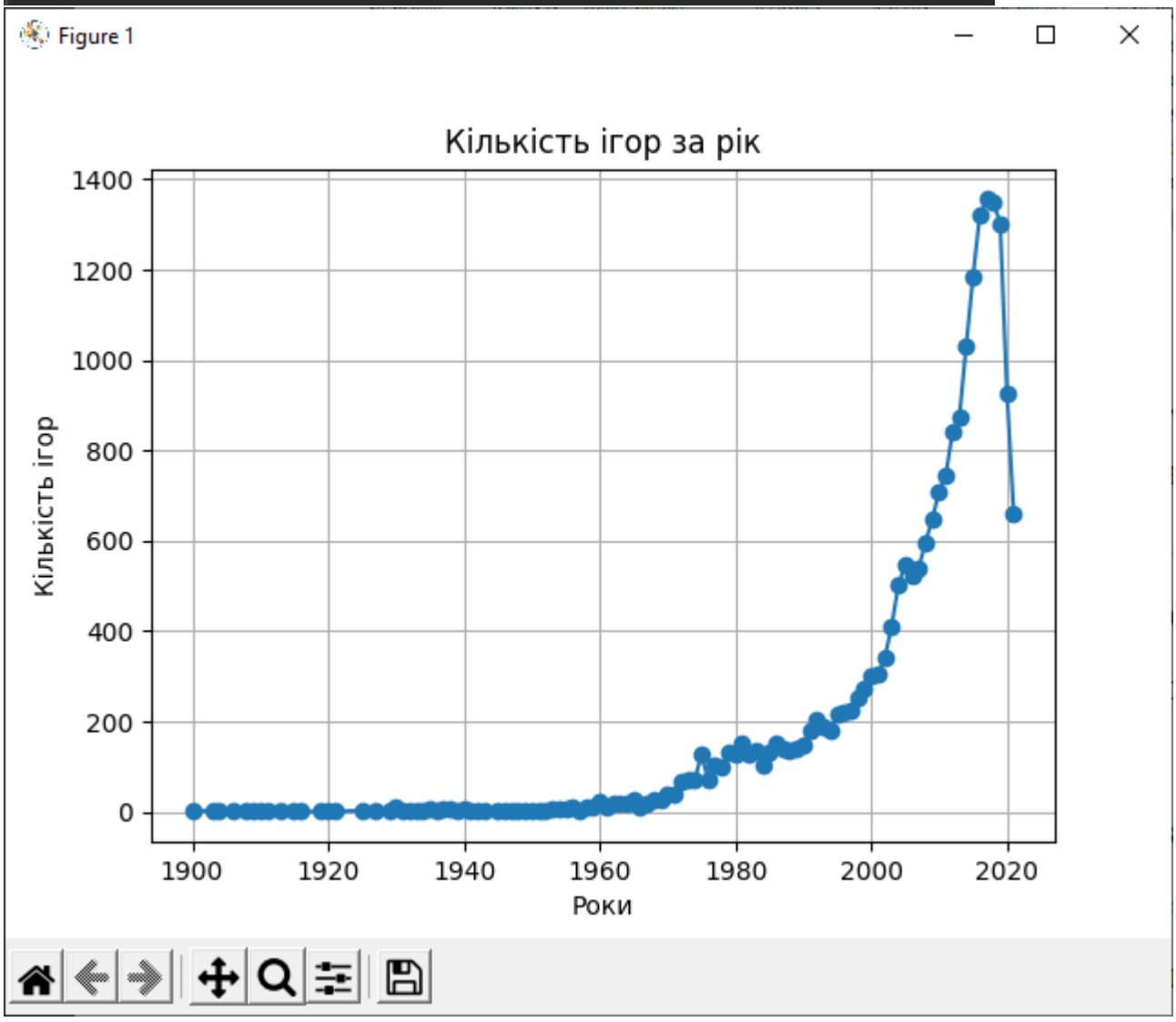

Рисунок 3.4 - виправлена кількість ігор випущених по роках

Тепер роки є коректними

Також ми бачимо що колонки, що відповідають за належність ігор до деякої категорії, представляють бінарні прапорці, 1 гра належить до категорії, 0 не належить.

Провсяк випадок, очищаємо дані від дуплікатів

```
dataset = dataset.drop_duplicates(subset=['Name', 'BGGId'])
```

Рисунок 3.5 – видалення дуплікатів

Виведемо перші 10 рядків

```
print("    Перші 10 рядків")
print(dataset.head(10))
```

```
    Перші 10 рядків
   BGGId                Name  AvgRating  YearPublished  GameWeight  \
0      1          Die Macher    7.61428           1986      4.3206
1      2        Dragonmaster    6.64537           1981      1.9630
2      3             Samurai    7.45601           1998      2.4859
3      4      Tal der Könige    6.60006           1992      2.6667
4      5             Acquire    7.33861           1964      2.5031
5      6  Mare Mediterraneum    6.55370           1989      3.0000
6      7           Cathedral    6.52083           1978      1.7950
7      8   Lords of Creation    6.10716           1993      2.4000
8      9        El Caballero    6.45265           1998      3.1824
9     10           Elfenland    6.69695           1998      2.1578

   Cat_Thematic  Cat_Strategy  Cat_War  Cat_Family  Cat_CGS  Cat_Abstract  \
0             0             1        0           0        0             0
1             0             1        0           0        0             0
2             0             1        0           0        0             0
3             0             0        0           0        0             0
4             0             1        0           0        0             0
5             0             0        0           0        0             0
6             0             0        0           0        0             1
7             0             0        0           0        0             0
8             0             1        0           0        0             0
9             0             0        0           1        0             0

   Cat_Party
0          0
1          0
2          0
3          0
4          0
5          0
6          0
7          0
8          0
9          0
```

Рисунок 3.6 – перші 10 рядків датафрейму

Після огляду даних та очистки від аномальних даних, ми можемо підготовити дані для розділення на тестувальну та тренувальні вибірки

Для цього спочатку відділяємо потрібні колонки і розділяємо дані на на вхідні ознаки та цільову змінну

```
X = dataset[['YearPublished', 'GameWeight', 'Cat_Thematic', 'Cat_Strategy', 'Cat_War',
             'Cat_Family', 'Cat_CGS', 'Cat_Abstract', 'Cat_Party']]
y = dataset['AvgRating']
```

Рисунок 3.7 – розподіл даних на вхідні ознаки та цільову змінну

Далі ми генеруємо поліноміальні ознаки для масиву X, щоб покращити результат прогнозування

```
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
```

Рисунок 3.8 – генерація поліноміальних ознак

Тепер ми можемо розділити наші дані на тренувальну та тестувальну вибірки. Розділяти ми будемо у відношенню 80% на 20%. Це треба бля того що б перевірити моделі на тестових, або невидимих даних.

```
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)
```

Рисунок 3.9 – розділення даних на вибірки

4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу даних

Для розвязання задачі з прогнозування рейтингу настільних ігор я обрав регресивні моделі LinearRegression, DecisionTreeRegressor та RandomForestRegressor. Кожна з цих моделей має свої плюси, які можуть бути корисні для прогнозування рейтингу гри на основі набору ознак.

LinearRegression є простою моделлю, яка дозволяє мені розуміти вплив кожної ознаки на прогнозований рейтинг гри. Це дуже корисно, оскільки я можу вивчити, які фактори мають найбільший вплив на успішність гри. Крім того, RandomForestRegressor досить проста для використання, і вона швидко навчається, що дозволяє мені ефективно працювати з даними.

DecisionTreeRegressor є гнучкою моделлю, яка може моделювати складні нелінійні залежності між ознаками і рейтингом гри. Це дозволяє мені автоматично виявити і використовувати важливі фактори для прогнозування. Крім того, DecisionTreeRegressor легко інтерпретується, що означає, що я можу зрозуміти, як модель приймає рішення на основі ознак.

RandomForestRegressor є потужною моделлю, яка поєднує кілька рішучих дерев для отримання високої точності прогнозування. Це особливо корисно, оскільки я хочу отримати якнайточніший прогноз рейтингу гри. Крім того, RandomForestRegressor стійкий до перенавчання і шуму в даних, а також може оцінювати важливість кожної ознаки, що допомагає мені виокремити ключові фактори, що впливають на рейтинг гри.

4.2 Аналіз отриманих результатів для методу LinearRegression

Тепер перейдемо до створення регресивних моделей. Після навчання моделі на тренувальних даних (X_train, y_train), я використав її для прогнозування рейтингу на тестових даних (X_test). Потім використав дві метри для перевірки якості прогнозування - Mean Squared Error (MSE) та R^2 Score. MSE вимірює середньо-квадратичну помилку між прогнозованими і фактичними

значеннями рейтингу, а R^2 Score вимірює відповідність прогнозованих значень до фактичних.

```
lin = LinearRegression()

lin.fit(X_train, y_train)

y_pred_lin = lin.predict(X_test)

mse = mean_squared_error(y_test, y_pred_lin)
r2 = r2_score(y_test, y_pred_lin)
```

```
Лінійна регресія
Mean Squared Error: 0.47521495538757347
R^2 Score: 0.43623786519919094
```

Рисунок 4.1 – тренування та результат прогнозування

Як ми бачимо оцінки Mean Squared Error (MSE) дорівнює 0.4752 та R^2 Score має значення 0.4362, що вказує на те, що модель може пояснити близько 43.62% варіації відповідей.

Загалом, оцінки якості прогнозування лінійної регресії вказують на помірний рівень точності моделі. Модель може давати розумні прогнози рейтингу настільних ігор, але її точність може бути поліпшена. Для цього я спробую використати більш складні моделі.

Також виведемо графік залежності справжніх даних від передбачених.

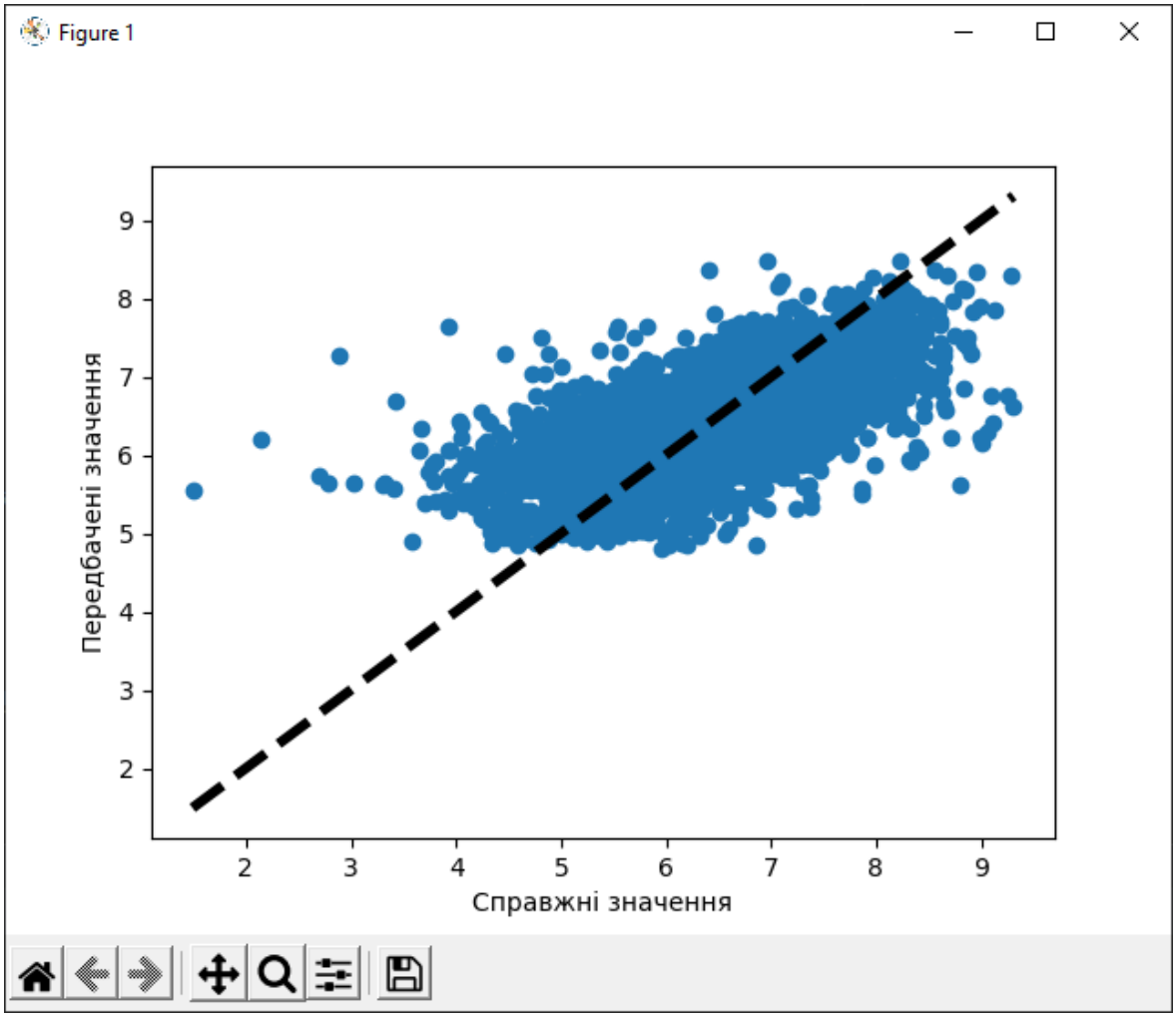

Рисунок 4.2 – графік залежності справжніх даних від передбачених лінійної регресії

На ньому ми бачимо що модель краще передбачає середні та високі оцінки гри ніж малі.

4.3 Аналіз отриманих результатів для методу DecisionTreeRegressor

Далі будуємо більш складну модель DecisionTreeRegressor. Спочатку я знов навчив модель на тренувальних даних, використовуючи функцію fit(), і потім зробив прогноз на тестових даних за допомогою функції predict(). Для оцінки якості прогнозування я використав ті самі дві метри - Mean Squared Error (MSE) і R^2 Score.

```
DRG = DecisionTreeRegressor()

DRG.fit(X_train, y_train)

y_pred_drg = DRG.predict(X_test)

mse = mean_squared_error(y_test, y_pred_drg)
r2 = r2_score(y_test, y_pred_drg)
```

```
Decision Tree Regression
Mean Squared Error: 0.6409986930882917
R^2 Score: 0.23956351221047123
```

Рисунок 4.3 – тренування та результат прогнозування

Оцінки Mean Squared Error (MSE) дорівнює 0.641 та R^2 Score має значення 0.240, що вказує на те, що модель може пояснити лише близько 24% варіації відповідей.

Відштовхуючись цих оцінок можна зробити висновок, що модель DecisionTreeRegressor показує менш задовільні результати у прогнозуванні рейтингу настільних ігор. Вона має високу середньо-квадратичну помилку і низький коефіцієнт детермінації, що свідчить про нестабільність та невисоку точність прогнозів.

Також будуємо графік залежності справжніх даних від передбачених.

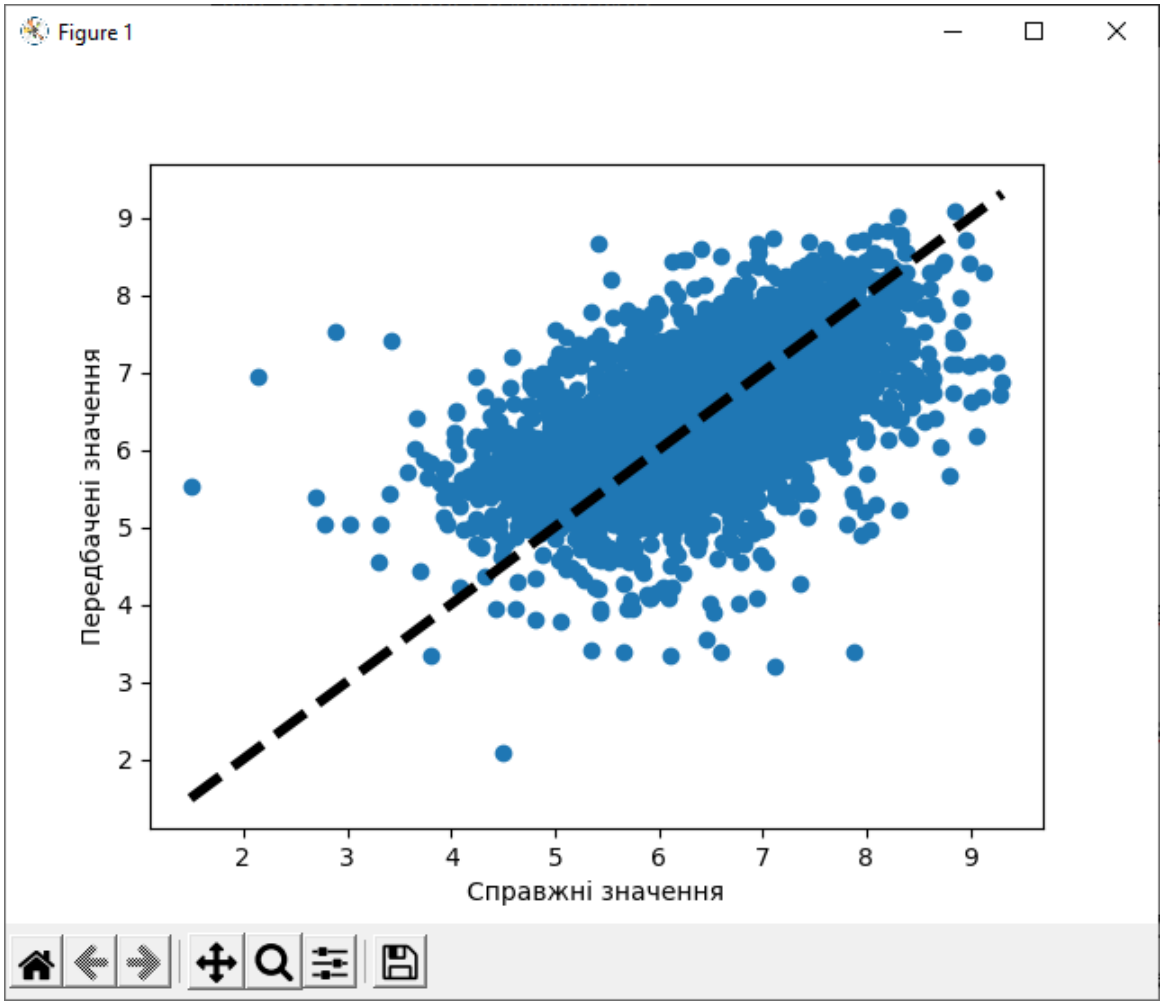

Рисунок 4.4 – графік залежності справжніх даних від передбачених для DecisionTreeRegressor

Як ми бачимо результати більш розкидані, що підтвержує низьку тсабільність.

4.4 Аналіз отриманих результатів для методу RandomForestRegressor

Тепер будуємо останню модель RandomForestRegressor. Під час створення моделі я використав random_state=20. Потім я навчив модель на тренувальних даних за допомогою функції fit() і зробив прогноз на тестових даних за допомогою функції predict().

```
RFR = RandomForestRegressor(random_state=20)

RFR.fit(X_train, y_train)

y_pred_RFR = RFR.predict(X_test)


mse = mean_squared_error(y_test, y_pred_RFR)
r2 = r2_score(y_test, y_pred_RFR)
```

```
Random Forest Regression
Mean Squared Error: 0.4950172872084149
R^2 Score: 0.41274574918982276
```

Рисунок 4.5 – тренування та результат прогнозування

За результатами ми отримали наступні оцінки якості прогнозування: Mean Squared Error (MSE) дорівнює 0.495 і R^2 Score становить 0.413.

За такими оцінками можна зробити висновок, що модель RandomForestRegressor показує прийнятну точність у прогнозуванні рейтингу настільних ігор. Хоча її середньо-квадратична помилка не є дуже низькою, модель все ж має здатність пояснювати певну частку варіації відповіді.

Також побудуємо графік

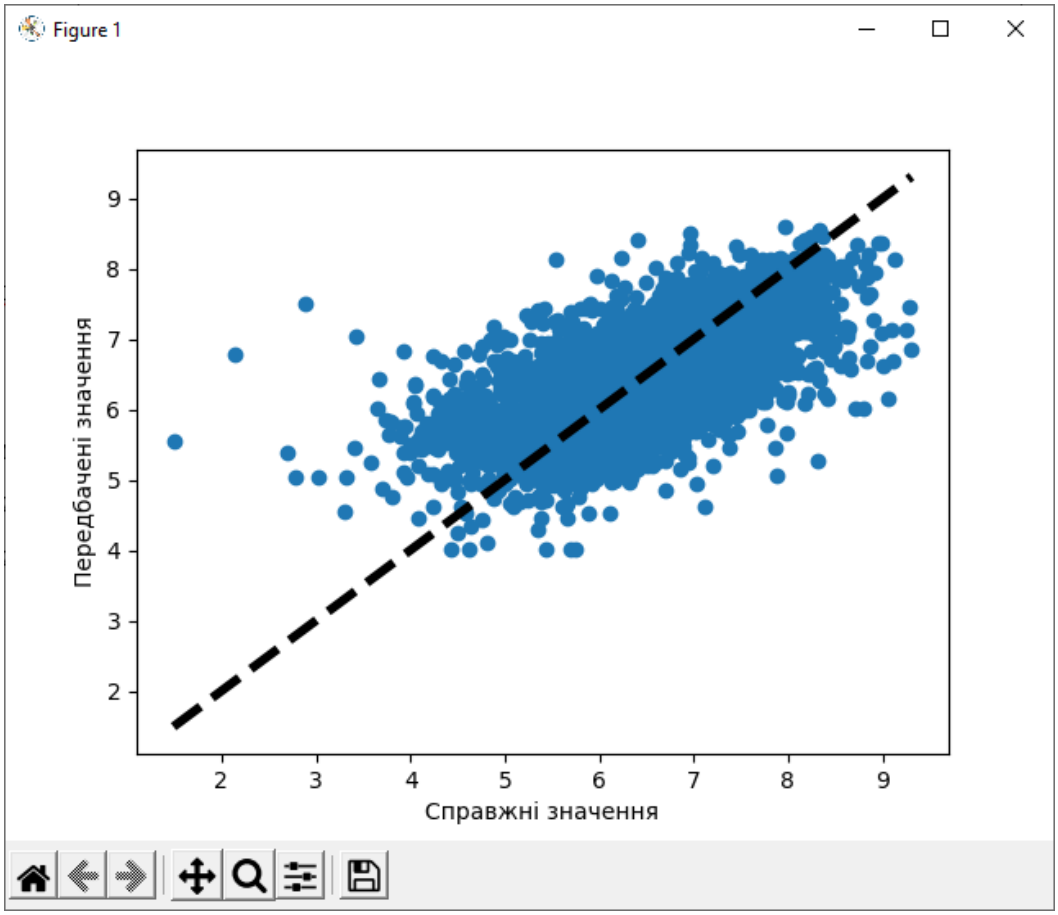

Рисунок 4.6 – графік залежності справжніх даних від передбачених для RandomForestRegressor

Як ми бачимо оцінки підтвердилися.

4.5 Порівняння ефективності методів інтелектуального аналізу

Для зручності оцінки зберемо всі оцінки разом

```
print("Порівняння R2 та MSE")
print("R2: \n\tLinearRegression: {0}\n\tDecision Tree Regression: {1}\n\tRandomForestRegressor: {2}"
      .format(r2_score(y_test, y_pred_lin), r2_score(y_test, y_pred_drg), r2_score(y_test, y_pred_RFR)))
print("MSE: \n\tLinearRegression: {0}\n\tDecision Tree Regression: {1}\n\tRandomForestRegressor: {2}"
      .format(mean_squared_error(y_test, y_pred_lin), mean_squared_error(y_test, y_pred_drg), mean_squared_error(y_test, y_pred_RFR)))
```

```
Порівняння R2 та MSE
R2:
    LinearRegression: 0.436237865199190 94
    Decision Tree Regression: 0.23956351221047123
    RandomForestRegressor: 0.41274574918982276
MSE:
    LinearRegression: 0.475214955387573 47
    Decision Tree Regression: 0.64099869308829 17
    RandomForestRegressor: 0.4950172872084149
```

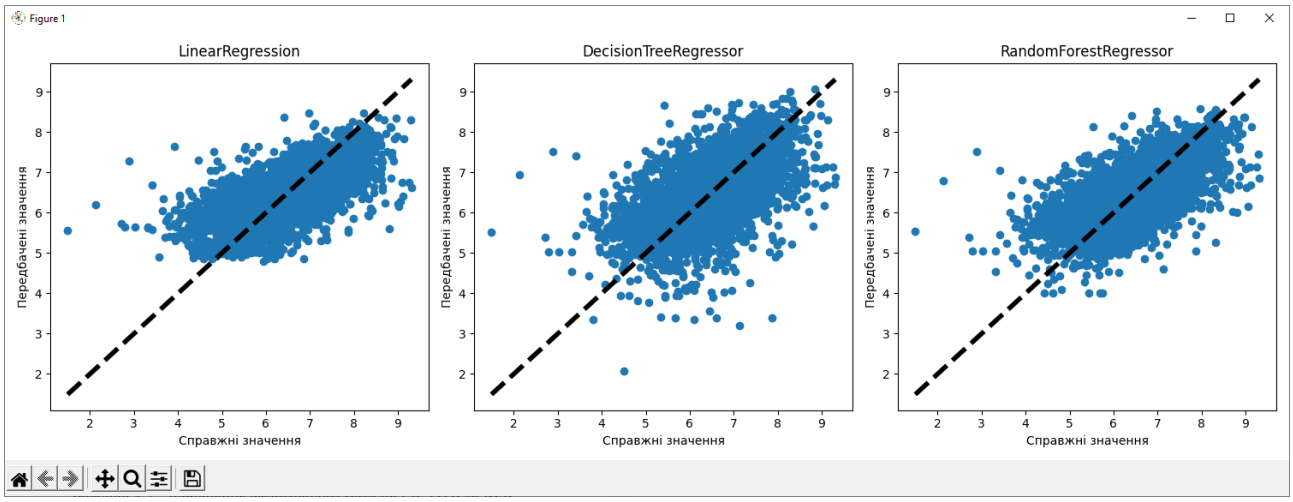

Рисунок 4.7 – порівняння ефективності методів LR, DTR та RFR

Тепер розберемо результати.

Дивлячись на коефіцієнт детермінації, ми бачимо, що модель LinearRegression має найвище значення R^2 (0.436), що означає, що дорівнює приблизно 43.6%. Модель Decision Tree Regression має значення R^2 (0.240), що є значно нижчим, що свідчить про меншу здатність моделі пояснювати варіацію.

Модель RandomForestRegressor показує проміжний результат зі значенням R^2 (0.413).

Далі проаналізуємо Mean Squared Error. Модель LinearRegression має найнижче значення MSE (0.475), що свідчить про меншу помилку в прогнозуванні. Модель Decision Tree Regression має більше значення MSE (0.641), вказуючи на більшу середньо-квадратичну помилку. Модель RandomForestRegressor показує проміжний результат зі значенням MSE (0.495).

Найбільшу ефективність показала LinearRegression, але вона всеодно має ефективність близька 50%, що не дає гарантії у прогнозуванні. Це можна пояснити тим, що рейтинг настільної ігри досить складна задача і залежить не тільки від параметрів самої гри. Багато чинників, які впливають на рейтинг, ми не можемо виміряти, наприклад моду на деякий стиль чи механіку або культурні чи регіональні особливості.

ВИСНОВКИ

У даній роботі ми вивчали можливість прогнозування рейтингу настільних ігор на основі різних параметрів. Для цього були використані три регресійні моделі: LinearRegression, DecisionTreeRegressor та RandomForestRegressor. Наша мета полягала у визначенні найбільш ефективної моделі для прогнозування рейтингу.

За результатами експериментів, отриманими з використанням набору даних, ми провели порівняння моделей за допомогою двох метрик: R^2 Score і Mean Squared Error (MSE).

Хоча модель LinearRegression показала найвищу ефективність серед усіх розглянутих моделей, результати все ж таки недостатньо задовільні. Це пояснюється тим, що прогнозування рейтингу настільних ігор є складною задачею і залежить від багатьох чинників, які не можуть бути повністю враховані в наборі параметрів. Наприклад, рейтинг може бути сильно вплинутий модою на певний стиль чи механіку гри, а також культурними чи регіональними особливостями.

Таким чином, навіть найкраща модель не може гарантувати точний прогноз рейтингу настільної гри. Однак результати нашої роботи вказують на те, що модель LinearRegression може бути корисною у визначенні загальних тенденцій та основних факторів, що впливають на рейтинг гри.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація мови програмування Python. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/
2. Бібліотека Pandas. [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/
3. Бібліотека Matplotlib. [Електронний ресурс] – Режим доступу до ресурсу: https://matplotlib.org/stable/
4. Бібліотека Sklearn. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/user_guide.html
5. LinearRegression in sklearn. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
6. DecisionTreeRegressor in sklearn. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
7. RandomForestRegressor in sklearn. [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Прогнозування рейтенгу настольної гри в залежності від параметрів. Методи Linearregression, Decisiontreeregressor, Randomforestregressor.

(Найменування програми (документа))

SSD

(Вид носія даних)

(Обсяг програми (документа), арк.,

студента групи ІП-*13 II курсу*

Дейнега В.М.

```
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
# Завантаження основного датасету
dataset = pd.read_csv('D:/Git/Kursova/data/games.csv')
dataset = dataset[['BGGId', 'Name', 'AvgRating', 'YearPublished',
'GameWeight', 'Cat_Thematic', 'Cat_Strategy', 'Cat_War',
                   'Cat_Family', 'Cat_CGS', 'Cat_Abstract', 'Cat_Party']]

# Аналіз данних
print("            Інформація")
print(dataset.info())
print()
print("    Опис")
print(dataset.describe())
print()
print("    Перші 10 рядків")
print(dataset.head(10))

year_info = dataset['YearPublished'].value_counts().sort_index()
plt.plot(year_info.index, year_info.values, marker='o')
plt.xlabel('Роки')
plt.ylabel('Кількість ігор')
plt.title('Кількість ігор за рік')
```

```
    plt.grid(True)
    plt.show()

    # Підготовка даних
    dataset = dataset[dataset['YearPublished'] >= 1900]

    year_info = dataset['YearPublished'].value_counts().sort_index()
    plt.plot(year_info.index, year_info.values, marker='o')
    plt.xlabel('Роки')
    plt.ylabel('Кількість ігор')
    plt.title('Кількість ігор за рік')
    plt.grid(True)
    plt.show()

    dataset = dataset.drop_duplicates(subset=['Name', 'BGGId'])

    # Розділення даних на вхідні ознаки (X) та цільову змінну (y)
    X = dataset[['YearPublished', 'GameWeight', 'Cat_Thematic',
'Cat_Strategy', 'Cat_War',
                 'Cat_Family', 'Cat_CGS', 'Cat_Abstract', 'Cat_Party']]
    y = dataset['AvgRating']

    poly_features = PolynomialFeatures(degree=2, include_bias=False)
    X_poly = poly_features.fit_transform(X)

    # Розділення даних на тренувальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(X_poly, y,
test_size=0.2, random_state=42)

    #----------------------------------------Лінійна регресія-------
----------------------------------------
    # Ініціалізація моделі лінійної регресії
    lin = LinearRegression()
```

```
    # Навчання моделі на тренувальних даних
    lin.fit(X_train, y_train)

    # Прогнозування на тестовому наборі
    y_pred_lin = lin.predict(X_test)

    # Оцінка моделі
    mse = mean_squared_error(y_test, y_pred_lin)
    r2 = r2_score(y_test, y_pred_lin)
    print('Лінійна регресія')
    print('Mean Squared Error:', mse)
    print('R^2 Score:', r2)

    fig, ax = plt.subplots()
    ax.scatter(y_test, y_pred_lin)
    ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'k--', lw=4)
    ax.set_xlabel('Справжні значення')
    ax.set_ylabel('Передбачені значення')
    plt.show()

    #----------------------------------------Decision Tree
Regression-----------------------------------------------
    # Ініціалізація моделі лінійної регресії
    DRG = DecisionTreeRegressor()

    # Навчання моделі на тренувальних даних
    DRG.fit(X_train, y_train)

    # Прогнозування на тестовому наборі
    y_pred_drg = DRG.predict(X_test)

    # Оцінка моделі
    mse = mean_squared_error(y_test, y_pred_drg)
```

```
    r2 = r2_score(y_test, y_pred_drg)
    print('Decision Tree Regression')
    print('Mean Squared Error:', mse)
    print('R^2 Score:', r2)

    # Вивід графіку
    fig, ax = plt.subplots()
    ax.scatter(y_test, y_pred_drg)
    ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'k--', lw=4)
    ax.set_xlabel('Справжні значення')
    ax.set_ylabel('Передбачені значення')
    plt.show()

    #-----------------------------------------Random Forest
Regression-----------------------------------------
    # Ініціалізація моделі лінійної регресії
    RFR = RandomForestRegressor(random_state=20)

    # Навчання моделі на тренувальних даних
    RFR.fit(X_train, y_train)

    # Прогнозування на тестовому наборі
    y_pred_RFR = RFR.predict(X_test)

    # Оцінка моделі
    mse = mean_squared_error(y_test, y_pred_RFR)
    r2 = r2_score(y_test, y_pred_RFR)
    print('Random Forest Regression')
    print('Mean Squared Error:', mse)
    print('R^2 Score:', r2)

    # Вивід графіку
    fig, ax = plt.subplots()
```

```
    ax.scatter(y_test, y_pred_RFR)
    ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'k--', lw=4)
    ax.set_xlabel('Справжні значення')
    ax.set_ylabel('Передбачені значення')
    plt.show()

    #----------------------------------------Загальні результати----
----------------------------------------
    print("Порівняння R2 та MSE")
    print("R2: \n\tLinearRegression: {0}\n\tDecision Tree Regression:
{1}\n\tRandomForestRegressor: {2}"
          .format(r2_score(y_test, y_pred_lin), r2_score(y_test,
y_pred_drg), r2_score(y_test, y_pred_RFR)))
    print("MSE: \n\tLinearRegression: {0}\n\tDecision Tree Regression:
{1}\n\tRandomForestRegressor: {2}"
          .format(mean_squared_error(y_test, y_pred_lin),
mean_squared_error(y_test, y_pred_drg), mean_squared_error(y_test,
y_pred_RFR)))

    fig, ax = plt.subplots(1, 3, figsize=(15, 5))

    ax[0].scatter(y_test, y_pred_lin)
    ax[0].plot([y_test.min(), y_test.max()], [y_test.min(),
y_test.max()], 'k--', lw=4)
    ax[0].set_xlabel('Справжні значення')
    ax[0].set_ylabel('Передбачені значення')
    ax[0].set_title('LinearRegression')

    ax[1].scatter(y_test, y_pred_drg)
    ax[1].plot([y_test.min(), y_test.max()], [y_test.min(),
y_test.max()], 'k--', lw=4)
    ax[1].set_xlabel('Справжні значення')
    ax[1].set_ylabel('Передбачені значення')
```

```
      ax[1].set_title('DecisionTreeRegressor')

      ax[2].scatter(y_test, y_pred_RFR)
      ax[2].plot([y_test.min(), y_test.max()], [y_test.min(),
y_test.max()], 'k--', lw=4)
      ax[2].set_xlabel('Справжні значення')
      ax[2].set_ylabel('Передбачені значення')
      ax[2].set_title('RandomForestRegressor')

      plt.tight_layout()
      plt.show()
```