Міністерство освіти і науки України
Національний технічний університет України
«Київський Політехнічний Інститут ім. Ігоря
Сікорського »

Кафедра прикладної математики

Пояснювальна записка до розрахунково-графічної роботи

з дисципліни «Математичне моделювання»

Тема: «Прогнозування рейтингу фільмів»

Виконала:
студентка групи КМ-83
Анфілова Олександра

Перевірив викладач:
Норкін Б.В.

ВСТУП

1. Постановка задачі

Предметною областю даного дослідження є сфера кінематографу. Адже у світі немає такої людина, яка б не любила дивитись фільми. А от які саме користуються найбільшою популярністю та мають найвищий рейтинг і доведеться дізнатись.

Основна задача - це прогнозування рейтингу фільмів на основі доступної інформації.(Акторський склад, режисер, кількість голосів, країна тощо).

Результати дослідження будуть корисними для великих медійних каналів, що виробляють ТВ- і відеоматеріали , і яким необхідно знати, чи сподобається глядачам той чи інший фільм.

2. Огляд літератури

1) The Elements of Statistical Learning Data Mining/ Inference/and Prediction/Trevor Hastie Robert Tibshirani Jerome Friedman
2) /python-data-science-machine-learning-tutorial/
3) Python Machine Learning/Себастьян Рашка, Вахід Мірджалілі/2019/ст.90-100
4) регрессия Лассо — DATA SCIENCE
5) Lasso Regression: Simple Definition - Statistics How To

3. ПРОЕКТУВАННЯ МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Перелік методів розв’язання задачі

Основна задача - це прогнозування рейтингу фільмів на основі доступної інформації.(Акторський склад, режисер, кількість голосів, країна тощо). З точки зору статистики та машинного навчання, задача прогнозування є задачею регресії. Для розв'язання задачі регресії необхідно змоделювати взаємоз'язок між змінними (залежними та незалежними). Система повинна прогнозувати популярність картини, тож необхідно знайти зв'язок між його параметрами.

Для розв'язання задачі регресії розроблені алгоритми спеціальні алгоритми:

-Лінійна Регресія

-Лассо-регресія

-Логістична Регресія

-Поліноміальна Регресія

-Дерева Прийняття Рішень

Лінійна регресія будує логістичну криву, яка вираховує ймовірність виникнення певної події, саме тому вона більше підходить для задач класифікації. З плюсів можна виділити швидке моделювання, а серед мінусів - складність будування поліноміальних моделей.

Лассо-регресія - ще аналог лінійної регресії. Дозволяє позбутися від зайвих незалежних змінних, які не впливають на залежну змінну.

Логістична регресія використовується, коли залежна змінна є двійковою, тому цей метод більше підходить до задач класифікації, ніж до задачі регресії.

Поліноміальна регресія будує криву довільного порядку (квадратичну, кубічну і тд).

Дерево рішень - простий для розуміння алгоритм. Щоб змоделювати значення залежної змінної, перевіряються різні умови, які за своєю структурою нагадують дерево. Кінцевим результатом є дерево із вузлами прийняття рішень (decision nodes) та листя (leafs). На ребрах («гілках»)

дерева рішення записані ознаки, від яких залежить цільова функція, а в «листі» записані значення цільової функції, а в інших вузлах - ознаки, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення.

3.2 Математичний метод розв’язання задачі

Для вирішення даної задачі, було обрано метод лассо-регресії**.**

Лассо-регресія - це аналог лінійної регресії. Дозволяє позбутися від зайвих незалежних змінних, які не впливають на залежну змінну.

3.3 Контрольний приклад

В якості прикладу використано датасет, який був завантажений на веб-сайті Kaggle. Він представляє собою набір даних із 1000 найпопулярніших фільмів на IMDB за останні 10 років.

На першому етапі відбувається збір сирих даних та збереження ресурсів даних:

1. Підключення бібліотек для полегшення роботи з даними.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

2. Імпорт даних.

```
data = pd.read_csv("D:\Kaggle\IMDB-Movie-Data.csv"
```

3. Визначення структури даних.

```
995   996     Secret in Their Eyes  ...                NaN      45.0
996   997           Hostel: Part II  ...              17.54      46.0
997   998   Step Up 2: The Streets   ...              58.01      50.0
998   999              Search Party  ...                NaN      22.0
999  1000                Nine Lives  ...              19.64      11.0

[1000 rows x 12 columns]
```

Наступним етапом є Візуалізація (елементарний опис зібраного набору даних)

Розглядається детальний вміст датасету, а саме його розмірність (кількість записів та кількість колонок) та характеристики записів.

```
print(data.info())
```

```
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Rank                1000 non-null   int64
 1   Title               1000 non-null   object
 2   Genre               1000 non-null   object
 3   Description         1000 non-null   object
 4   Director            1000 non-null   object
 5   Actors              1000 non-null   object
 6   Year                1000 non-null   int64
 7   Runtime (Minutes)   1000 non-null   int64
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64
 10  Revenue (Millions)  872 non-null    float64
 11  Metascore           936 non-null    float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
None
```

З опису даного датасету на сторінці Kaggle можна дізнатись вміст усіх колонок датасету:

Title - Заголовок

Genre - Жанр

Description - Опис

Director - Режисер

Actors - Актори

Year - Рік

Runtime - Тривалість

Rating - Рейтинг

Votes - Голоси

Revenue - Прибуток

Metascrore - Оцінка критиків

Первинна очистка даних

Видалення поля “Description”, адже воно не має ніякого сенсу для дослідження.

```
data_new = data.drop(columns="Description")
```

```
print(data_new)
```

Наступним кроком буде усунення NaN, Null значень:

```
print(data_new.isnull().sum())
```

```
None
Rank                      0
Title                     0
Genre                     0
Director                  0
Actors                    0
Year                      0
Runtime (Minutes)         0
Rating                    0
Votes                     0
Revenue (Millions)      128
Metascore                64
dtype: int64
```

```
data_new.dropna(subset=['Revenue (Millions)', 'Metascore'],
inplace=True)
```

```
print(data_new)
```

```
None
Rank                    0
Title                   0
Genre                   0
Director                0
Actors                  0
Year                    0
Runtime (Minutes)       0
Rating                  0
Votes                   0
Revenue (Millions)      0
Metascore               0
dtype: int64
```

Далі відбувається перевірка на наявність записів-дублікати:

```
print(data_new.duplicated().sum())
```

```
None
0
```

Записи-дублікати в датасеті відсутні.

```
     Rank                     Title  ... Revenue (Millions) Metascore
0       1   Guardians of the Galaxy  ...              333.13      76.0
1       2                Prometheus  ...              126.46      65.0
2       3                     Split  ...              138.12      62.0
3       4                      Sing  ...              270.32      59.0
4       5             Suicide Squad  ...              325.02      40.0
..    ...                       ...  ...                 ...       ...
993   994  Resident Evil: Afterlife  ...               60.13      37.0
994   995                Project X  ...               54.72      48.0
996   997          Hostel: Part II  ...               17.54      46.0
997   998    Step Up 2: The Streets  ...               58.01      50.0
999  1000                Nine Lives  ...               19.64      11.0
```

Розвідувальний аналіз даних (EDA)

Застосовується описова статистика: пошук кількості non-NA/null значень, середнього арифметичного, стандартних відхилень, мінімального та максимального значення кожної характеристики за допомогою функції describe():

print(data_new.describe())

```
None
              Rank         Year  ...  Revenue (Millions)   Metascore
count   838.000000   838.00000  ...          838.000000  838.000000
mean    485.247017  2012.50716  ...           84.564558   59.575179
std     286.572065     3.17236  ...          104.520227   16.952416
min       1.000000  2006.00000  ...            0.000000   11.000000
25%     238.250000  2010.00000  ...           13.967500   47.000000
50%     475.500000  2013.00000  ...           48.150000   60.000000
75%     729.750000  2015.00000  ...          116.800000   72.000000
max    1000.000000  2016.00000  ...          936.630000  100.000000
```

Додатково відбувається пошук кожної характеристики. Для цього використовується вбудована в бібліотеку pandas функцією var().

```
print(data_new.var())
```

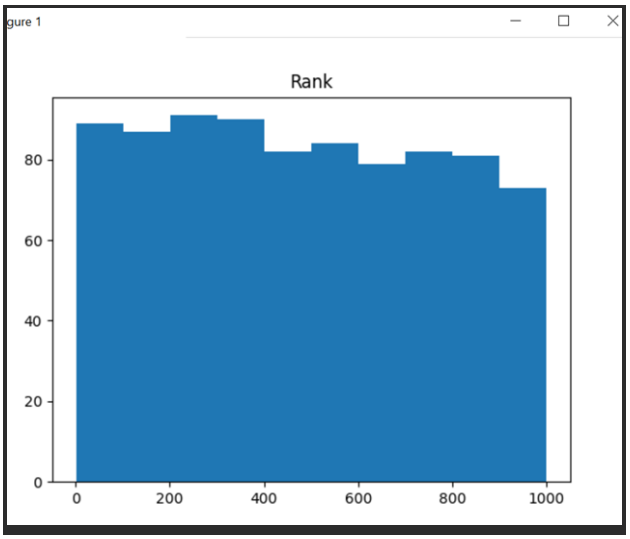
```
Rank                    8.212355e+04
Year                    1.006387e+01
Runtime (Minutes)       3.411750e+02
Rating                  7.704518e-01
Votes                   3.728723e+10
Revenue (Millions)      1.092448e+04
Metascore               2.873844e+02
dtype: float64
```

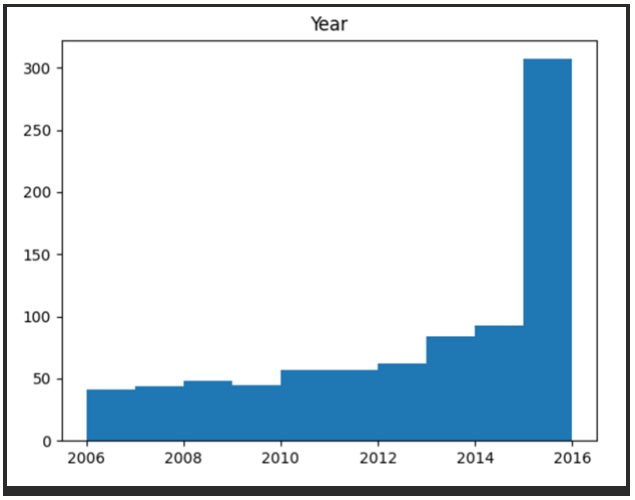
Пошук медіани, за допомогою вбудованої в бібілотеку pandas функції median().

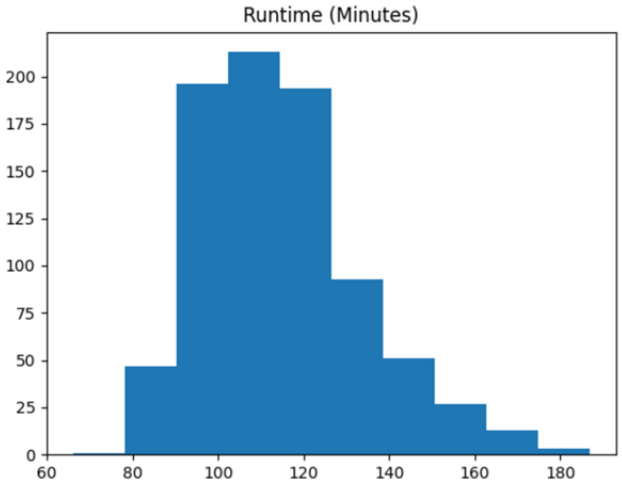
```
print(data_new.median())
```

Побудова гістограми розподілів числових ознак датасету. Для побудови графіків використовується бібліотека matplotlib:

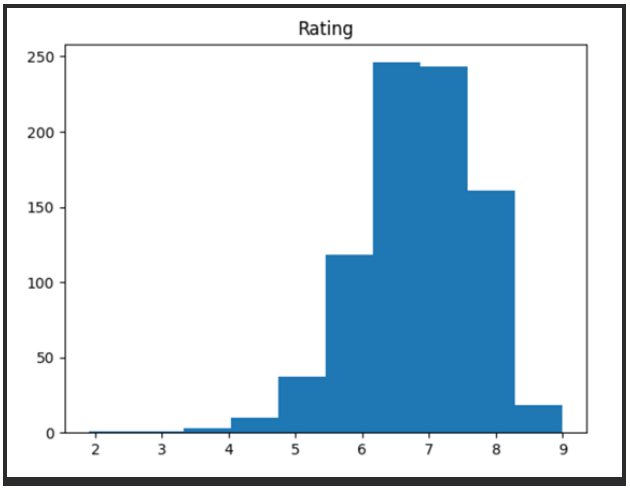
```
data_new_numeric = data_new.select_dtypes(include=[np.number])

numeric = data_new_numeric.columns.values

for i in numeric:

    plt.hist(data_new[i])

    plt.title(i)

    plt.show()
```

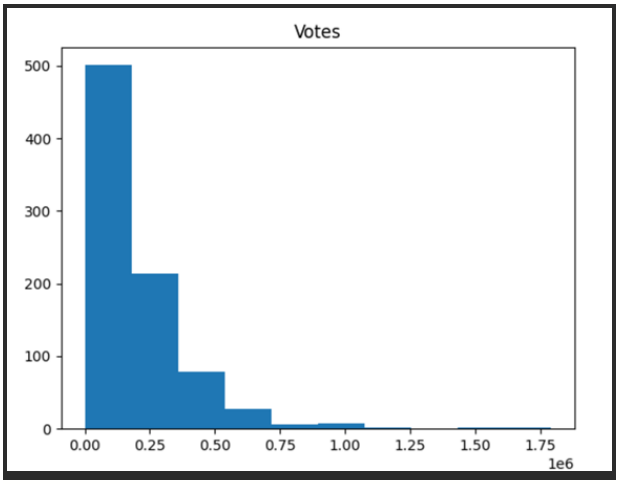


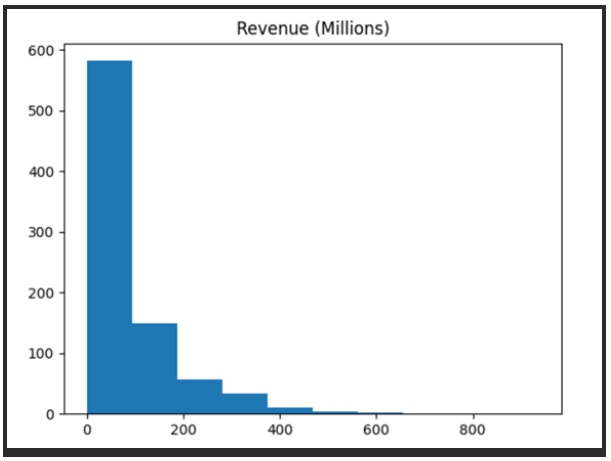


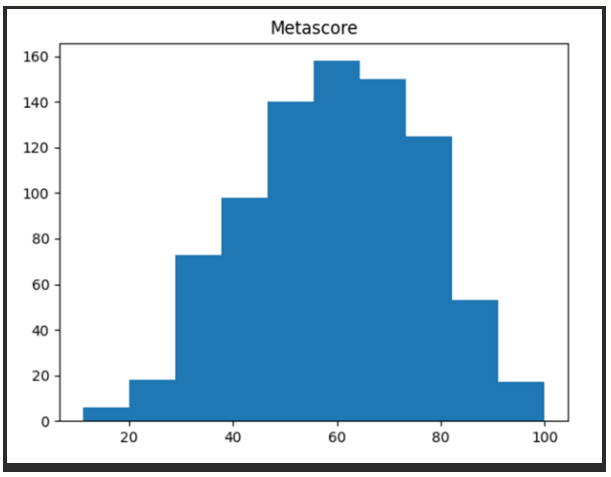



```
None
Rank                        475.50
Year                       2013.00
Runtime (Minutes)           112.00
Rating                        6.90
Votes                    136879.50
Revenue (Millions)          48.15
Metascore                   60.00
dtype: float64
```



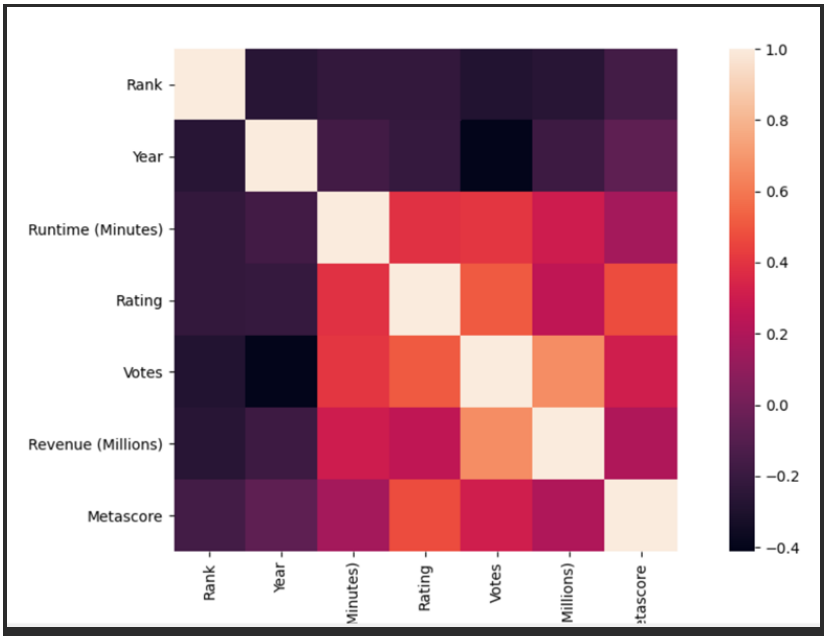






Далі проводиться кореляційний аналіз. За допомогою вбудованої в pandas функції corr(), будується кореляційна матриця числових ознак датасету

```
corr = data.corr()

sns.heatmap(corr, square=True)

plt.show()
```



Результати доводять, що між деякими ознаками є досить сильний взаємозв'язок. Найнижча кореляція між "кількістю голосів" та "роком випуску". "Кількість голосів" та "дохід" сильно корелюють між собою.

Розбиття dataset на тренувальний, валідаційний та тестовий набори даних

Розбиття відбувається за допомогою функції train_test_split().

```
X = data_new[data_new.columns[:-1]]

y = data_new[data_new.columns[-1]]

X_train, X_test, y_train, y_test = train_test_split(X, y)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train)


print(f'Train: X = {X_train.head()}, y = {y_train.head()}')

print(f'Test: X = {X_test.head()}, y = {y_test.head()}')

print(f'Validation: X = {X_val.head()}, y = {y_val.head()}')
```

```
Train: X =      Rank             Title  ...   Votes Revenue (Millions)
353   354      Café Society  ...   45579              11.08
384   385    The Lego Movie  ...  266508             257.76
639   640  American Reunion  ...  178471              56.72
137   138  The Great Gatsby  ...  386102             144.81
727   728   The Illusionist  ...  309934              39.83

[5 rows x 10 columns], y = 353    64.0
384    83.0
639    49.0
137    55.0
727    68.0
Name: Metascore, dtype: float64
```

```
Test: X =      Rank                        Title  ...   Votes Revenue (Millions)
135   136  The Place Beyond the Pines  ...  200090              21.38
67     68          Mad Max: Fury Road  ...  632842             153.63
594   595                Daddy's Home  ...   68306             150.32
86     87               Live by Night  ...   27869              10.38
6       7                  La La Land  ...  258682             151.06

[5 rows x 10 columns], y = 135    68.0
67     90.0
594    42.0
86     49.0
6      93.0
Name: Metascore, dtype: float64
```

```
Validation: X =      Rank                                    Title  ...   Votes Revenue (Millions)
372   373                                 Criminal  ...   38430              14.27
10     11  Fantastic Beasts and Where to Find Them  ...  232072             234.02
357   358               The Theory of Everything  ...  299718              35.89
727   728                       The Illusionist  ...  309934              39.83
268   269               X-Men Origins: Wolverine  ...  388447             179.88

[5 rows x 10 columns], y = 372    36.0
10     66.0
357    72.0
727    68.0
268    40.0
Name: Metascore, dtype: float64
```

Формалізація цільової функції оптимізації. Визначення метрик оцінки ефективності моделі

Нехай X - матриця, стовпчики якої - це значення незалежних змінних, а рядки - це фільми:

Цільова функція оптимізації :

$$loss(B) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - X_iB)^2 + \alpha\sum_{k=1}^{K}|B_k|.$$

Метрика оцінки ефективності моделі -- коефіцієнт детермінації

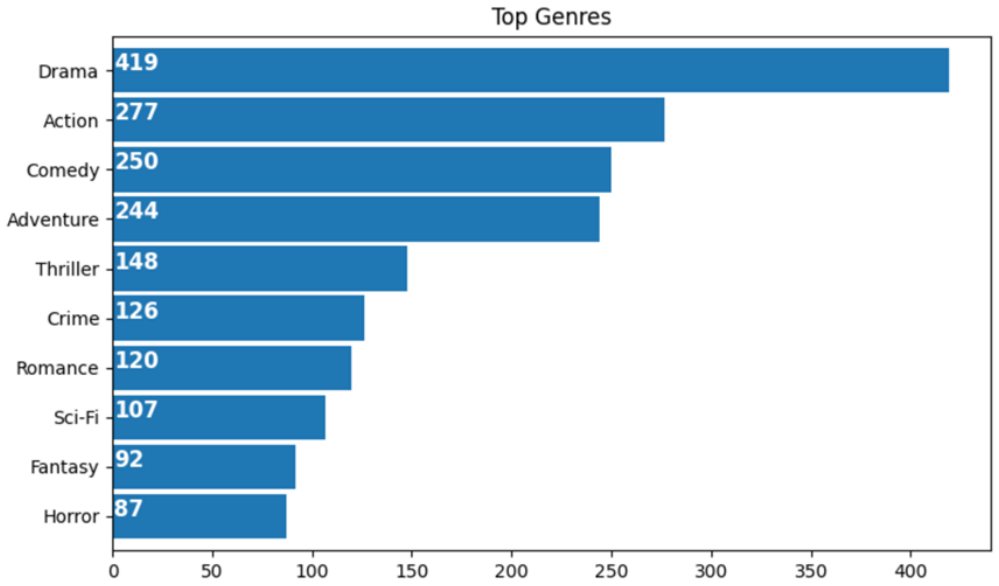
R^2. Він показує, яка частка загальної дисперсії даних пояснюється регресією. Формула: R^2 = $1 - \frac{\text{сума квадратів нев'язок}}{\text{загальна сума квадратів}}$

Робота з колонкою “Жанр”

Побудуємо діаграму, щоб побачити, які жанри є найпопулярнішими.

```
data_new['Genre'] =
data_new['Genre'].str.strip('[]').str.replace('
','').str.replace("'",'')

data_new['Genre'] = data_new['Genre'].str.split(',')

plt.subplots(figsize=(12,10))

list1 = []

for i in data_new['Genre']:

    list1.extend(i)

ax =
pd.Series(list1).value_counts()[:10].sort_values(ascending=Tru
e).plot.barh(width=0.9)
```

```
for i, v in
enumerate(pd.Series(list1).value_counts()[:10].sort_values(asc
ending=True).values):

    ax.text(.8, i, v,fontsize=12,color='white',weight='bold')

plt.title('Top Genres')

plt.show()
```



Наступним кроком є створення списку «genreList» із усіма можливими унікальними жанрами в наборі даних.

```
genreList = []

for index, row in data_new.iterrows():

    genres = row["Genre"]


    for genre in genres:

        if genre not in genreList:

            genreList.append(genre)
```

```
print(genreList[:10])
```

```
['Action', 'Adventure', 'Sci-Fi', 'Mystery', 'Horror', 'Thriller', 'Animation', 'Comedy', 'Family', 'Fantasy']
```

Створення нового стовпця, який буде містити двійкові значення, залежно від того, присутній у ньому жанр чи ні.

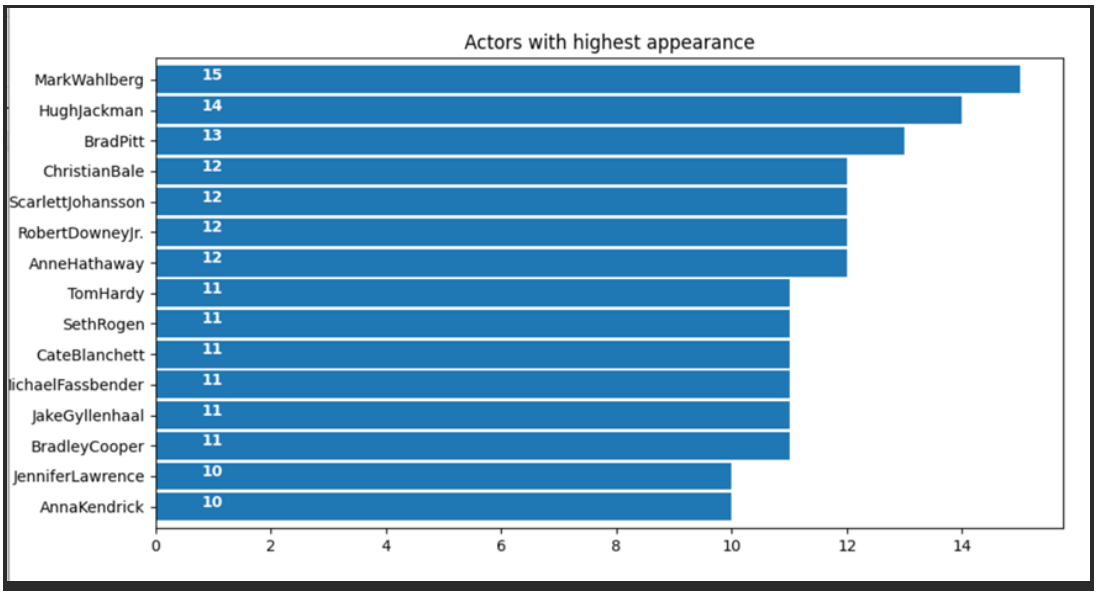
```
def binary(genre_list):
  binaryList = []
  for genre in genreList:
      if genre in genre_list:
          binaryList.append(1)
      else:
          binaryList.append(0)
  return binaryList
data_new['genres_bin'] = data_new['Genre'].apply(lambda x: binary(x))
print(data_new['genres_bin'].head())
```

```
0    [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1    [0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2    [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3    [0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, ...
4    [1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ...
Name: genres_bin, dtype: object
```

Далі працюємо із колонкою “Actors”

Побудуємо діаграму, щоб побачити, які актори зустрічаються найчастіше.

```
data_new['Actors'] =
data_new['Actors'].str.strip('[]').str.replace('
','').str.replace("'",'')

data_new['Actors'] = data_new['Actors'].str.split(',')

plt.subplots(figsize=(12,10))

list1=[]

for i in data_new['Actors']:

    list1.extend(i)

ax=pd.Series(list1).value_counts()[:15].sort_values(ascending=
True).plot.barh(width=0.9)

for i, v in
enumerate(pd.Series(list1).value_counts()[:15].sort_values(asc
ending=True).values):

    ax.text(.8, i, v,fontsize=10,color='white',weight='bold')

plt.title('Actors with highest appearance')

plt.show()
```



Марк Волберг, Г’ю Джекман та Бред Пітт є найпопулярнішими акторами.

Далі визначається внесок кожного актора у фільм:

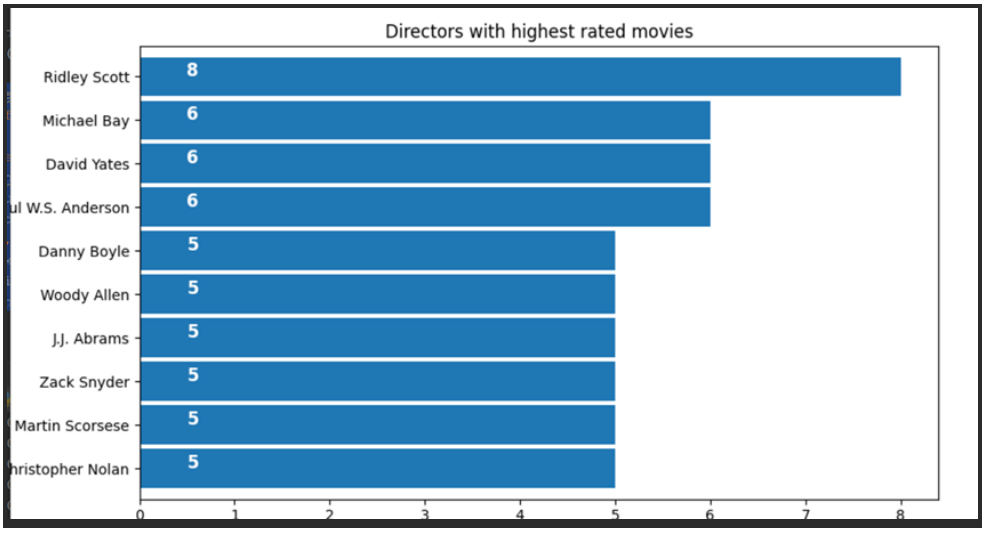
```
for i, j in zip(data_new['Actors'], data_new.index):
    list2 = []
    list2 = i[:4]
    data_new.loc[j, 'cast'] = str(list2)
for i, j in zip(data_new['Actors'], data_new.index):
    list2 = []
    list2 = i
    list2.sort()
    data_new.loc[j, 'Actors'] = str(list2)
castList = []
for index, row in data_new.iterrows():
    cast = row["Actors"]


    for i in cast:
        if i not in castList:
            castList.append(i)
data_new['cast_bin'] = data_new['Actors'].apply(lambda x:
binary(x))
print(data_new['cast_bin'].head())
```

```
0    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
4    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
Name: cast_bin, dtype: object
```

Робота з колонкою “Director”:

Побудова діаграми, щоб визначити, хто з режисерів створив найбільшу кількість картин.

```
def xstr(s):

    if s is None:

        return ''

    return str(s)

data_new['Director'] = data_new['Director'].apply(xstr)

plt.subplots(figsize=(12,10))

ax =
data_new[data_new['Director']!=''].Director.value_counts()[:10
].sort_values(ascending=True).plot.barh(width=0.9)

for i, v in
enumerate(data_new[data_new['Director']!=''].Director.value_co
unts()[:10].sort_values(ascending=True).values):

    ax.text(.5, i, v,fontsize=12,color='white',weight='bold')

plt.title('Directors with highest rated movies')

plt.show()
```



Створення колонки“director-bin”:

```
directorList=[]
```

```
for i in data_new['Director']:
```

```
    if i not in directorList:
```

```
        directorList.append(i)
```

```
data_new['director_bin'] = data_new['Director'].apply(lambda
x: binary(x))
```

```
print(data_new['director_bin'].head())
```

```
0    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
4    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
Name: director_bin, dtype: object
```

Математичне забезпечення

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn import metrics
```

```
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

from imblearn.over_sampling import SMOTE

import matplotlib.pyplot as plt

from sklearn.pipeline import make_pipeline

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.model_selection import KFold, cross_val_score,
train_test_split

from sklearn.metrics import mean_squared_error, r2_score
```

Імплементація обраного методу моделювання. Створення моделі. Стандартизація числових даних (feature scaling). Навчання моделі. Тестування навченної моделі

Для задачі лассо-регресії зручно використати вбудований модуль sklearn (scikit-learn). Це бібліотека Python, яка містить засоби для різноманітних задач машинного навчання: класифікації, регресії, кластеризації, зменшення розмірності простору (dimensionality reduction), вибору моделей тощо. За проведення лассо-регресії відповідає функція sklearn.linear_model.Lasso.

```
lasso = linear_model.Lasso(alpha=1, normalize=True)
```

За навчання моделі в бібліотеці sklearn відповідає функція fit. sklearn самостійно проведе регресію.

```
lasso.fit(X_train, y_train)

coefs = lasso.coef_

intercept = lasso.intercept

print("b_0 =", intercept)

pd.DataFrame(data=np.expand_dims(coefs, 0),
columns=X_train.columns)
```

```
b_0 = [-1302.2536382819]
```

Розглядається вільний член b_0 . Якщо прибуток, тривалість, оцінка критиків та інші характеристики дорівнюють нулю, то його рейтинг буде становити -1302.2536382819. Від’ємний рейтинг - це одна з особливостей лінійної регресії. Лінійна функція (якщо це не константа) в якійсь точці досягає нуля і набуває від'ємних значень.

Підрахунок коефіцієнту детермінації R, який чисельно показує, яка частина варіації залежної змінної пояснена моделлю:

```
r2_train = lasso.score(X_train, y_train)
r2_valid = lasso.score(X_val, y_val)
print(r2_train, r2_valid)
```

```
0.6318844061494144 0.5922100004904121
```

Для тренувального сету він дорівнює 0.632, а для валідаційного - 0.592. Тобто модель пояснює мінливість даних у датасеті в середньому на 62%.

Знайдемо для різних α модель та знайдемо коефіцієнт R^2. Оберемо ту модель, в якій цей коефіцієнт найбільший.

```
alpha_arr = [10000, 1000, 300, 200, 100, 30, 10, 3, 1, 0.3, 0.1, 0.03, 0.01]

models = []

for alpha in alpha_arr:

 lasso = linear_model.Lasso(alpha=alpha, normalize=True)

 lasso.fit(X_train, y_train)
```

```
models.append(lasso)
```

```
r2 = lasso.score(X_val, y_val)
```

```
print(f"alpha = {alpha}, R^2 = {r2}")
```

```
alpha = 10000, R^2 = 0.476650986785431
alpha = 1000, R^2 = 0.476650986785431
alpha = 300, R^2 = 0.476650986785431
alpha = 200, R^2 = 0.476650986785431
alpha = 100, R^2 = 0.712348953225844
alpha = 30, R^2 = 0.706650886785431
alpha = 10, R^2 = 0.69586335701235
alpha = 3, R^2 = 0.6866509867854318
alpha = 1, R^2 = 0.684650986787331
alpha = 0.1, R^2 = 0.68435098787331
alpha = 0.03, R^2 = 0.6821650216787331
alpha = 0.01, R^2 = 0.6811650986787331
```

Найвищу точність отримали при alpha=3.

Тепер візуально порівняємо істині показники рейтингу (Actual IMDb Rating) фільму та передбачені моделлю (Predicted IMDB Rating):

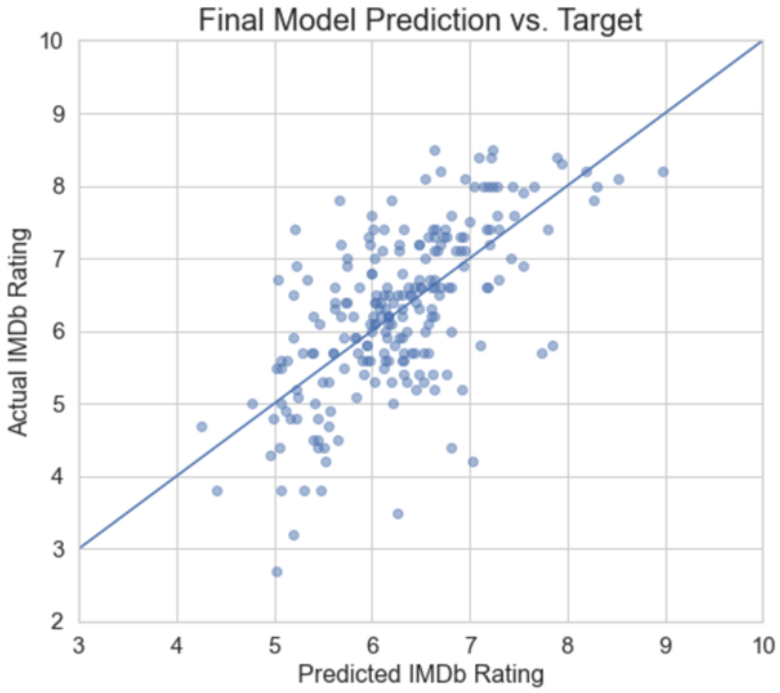
```
results = pd.DataFrame({'y_true': np.squeeze(Y_test),
'y_pred':Y_pred})
```

```
results.head(10)
```

```
y_truey_pred

7      6.9183739

4      4.46532182
```

```
5      4.8128364

8      8.345758

6      5.9433321

9      8.1032728

5      4.38755321

7      7.59974123

5      4.421097654

6      6.1209756753
```

Видно, що прогнозовані значення рейтингів є близькими до реальних значень. Але все-таки потрібно побудувати графік залежності прогнозованих оцінок і фактичного рейтингу.



З цього графіка видно, що в цілому модель зберігає тенденцію даних.

Але є певна категорія фільмів, які були сприйняті публікою неоднозначно, та при цьому мають високі рейтинги. В таких випадках важко спрогнозувати популярність картини.

3.4 Архітектура розроблюваних засобів

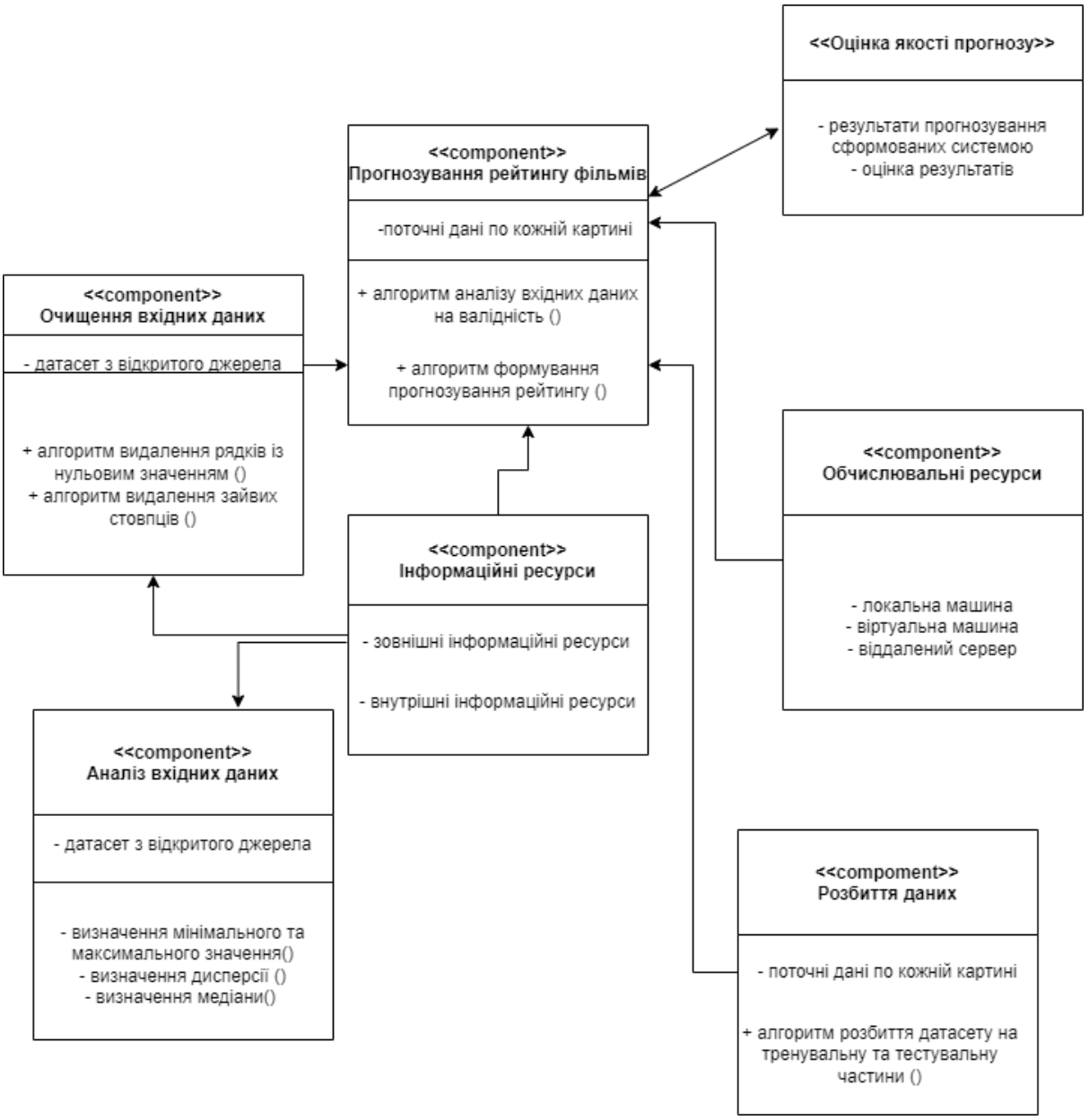


рис. 1 Архітектура програмного засобу

3.5 ФОРМАТ ВИХІДНИХ ДАНИХ

В якості інструментів взаємодії з вихідними даними було обрано мову програмування Python та особливо її бібліотеку pandas, що є досить слушним вибором через таргетованість даного стеку інструментів саме на роботу з великими об’ємами інформації

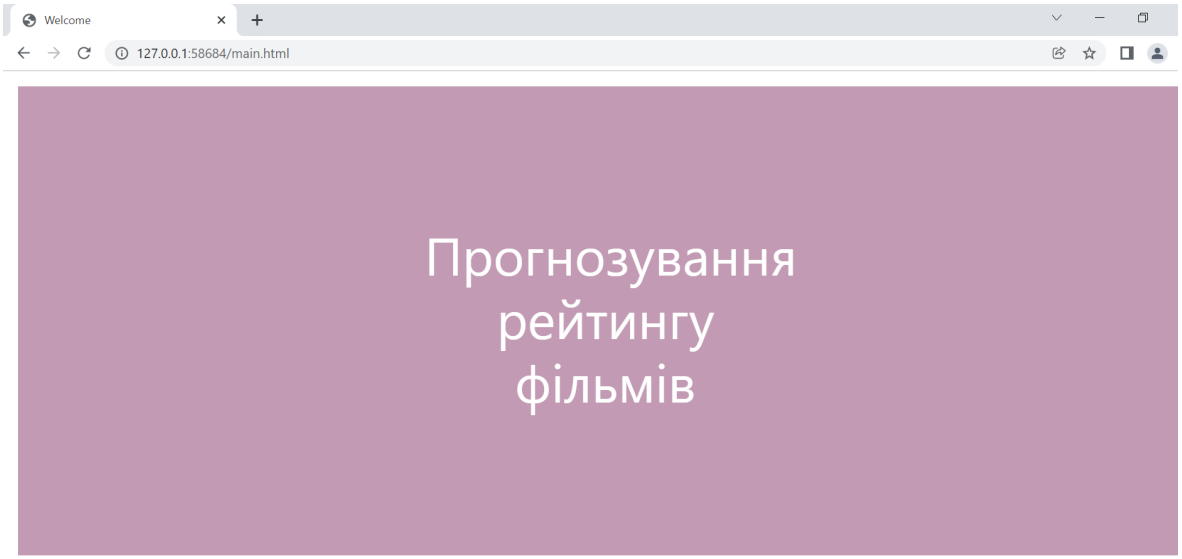
В якості вихідних даних було обрано масив даних з відкритого джерела який був завантажений на веб-сайті Kaggle. Він представляє собою набір даних із 1000 найпопулярніших фільмів на IMDB за останні 10 років.

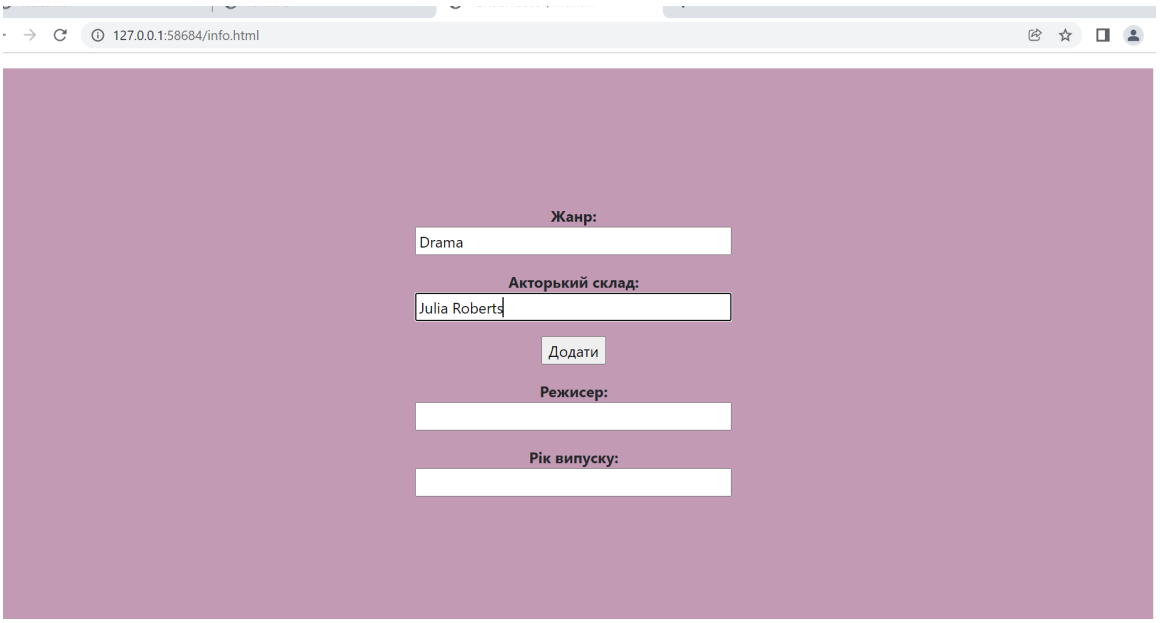
Слід зазначити, що, хоча сам датасет є цілком валідним, дані в ньому представлені правильно та повно.

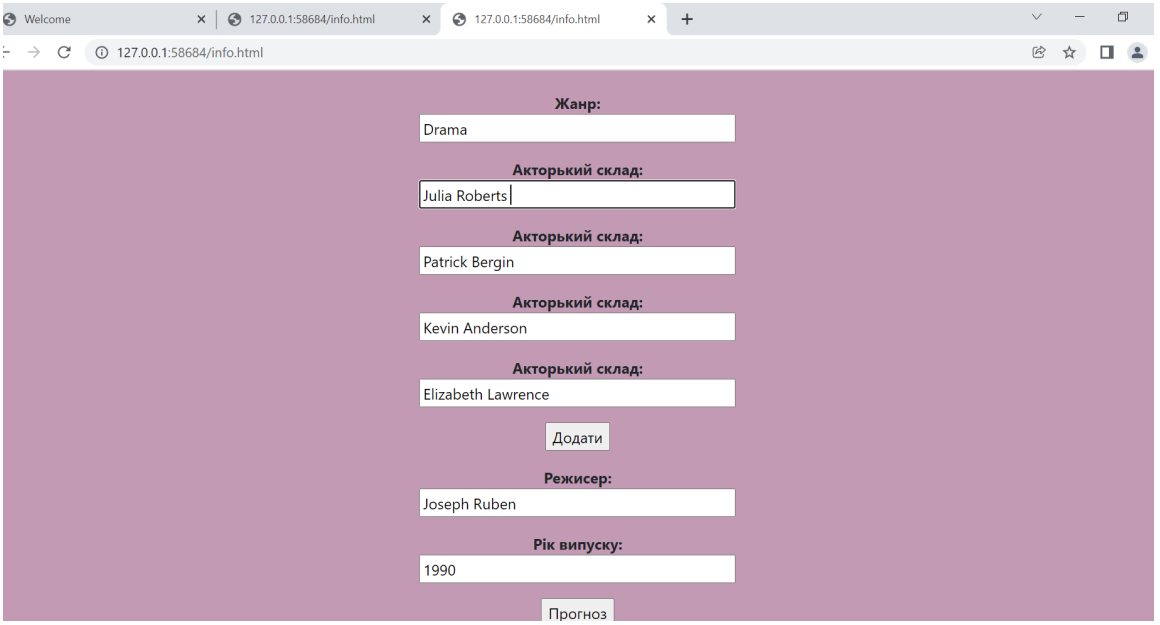
4. Розробка інтерфейсу користувача

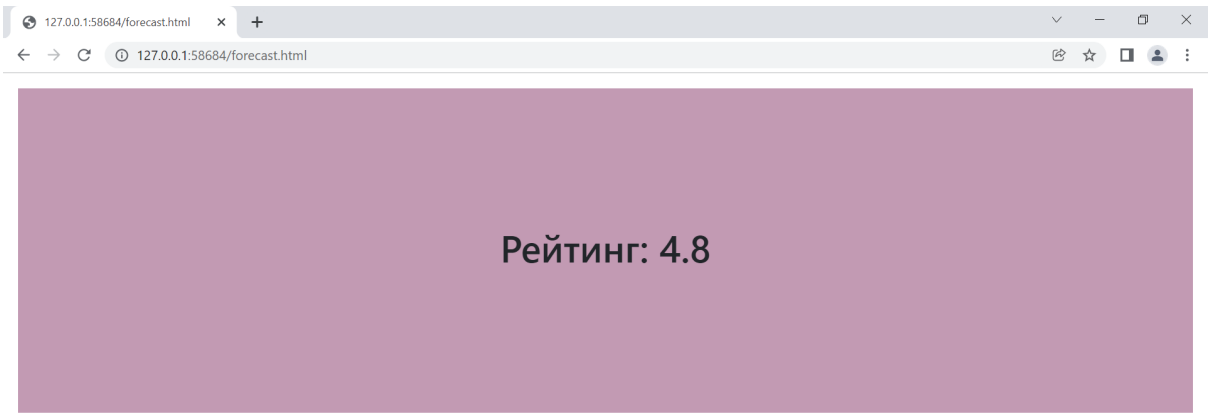
Для реалізації інтерфейсу для поставленої задачі було обрано мову програмування Python. Досить практичною є оболонка веб-застосунку. Для останнього обрано фреймворк Flask, а розмітку зроблено за допомогою фреймворку Bootstrap4.

Користувачеві відкривається головна сторінка




127.0.0.1:58684/info.html
Жанр:
Drama
Акторький склад:
Julia Roberts
Додати
Режисер:
Рік випуску:


Welcome
127.0.0.1:58684/info.html
127.0.0.1:58684/info.html
127.0.0.1:58684/info.html
Жанр:
Drama
Акторький склад:
Julia Roberts
Акторький склад:
Patrick Bergin
Акторький склад:
Kevin Anderson
Акторький склад:
Elizabeth Lawrence
Додати
Режисер:
Joseph Ruben
Рік випуску:
1990
Прогноз


127.0.0.1:58684/forecast.html
127.0.0.1:58684/forecast.html
Рейтинг: 4.8

Висновки

В результаті даної роботи було отримано модель. Основна ціль моделі - прогнозування рейтингу фільмів на основі доступної інформації, тобто, є задачею регресії. Було обрано метод лассо-регресія та на його основі побудовано модель.

Оцінено точність моделі: на тестовому сеті дана модель пояснює 62% усієї мінливості даних.

Для побудови моделі ми використали бібліотеку для мови Python - sklearn.

В ході даної роботи було проведено такі етапи:

-проаналізовано та обрано моделі для вирішення поставленої задачі;

-створено та навчено модель;

-інтерпретовано результати роботи моделі;

Результати дослідження будуть корисними для великих медійних каналів, що виробляють ТВ- і відеоматеріали , і яким необхідно знати, чи сподобається глядачам той чи інший фільм.

Література

6) The Elements of Statistical Learning Data Mining/ Inference/and Prediction/Trevor Hastie Robert Tibshirani Jerome Friedman
7) /python-data-science-machine-learning-tutorial/
8) Python Machine Learning/Себастьян Рашка, Вахід Мірджалілі/2019/ст.90-100
9) регрессия Лассо — DATA SCIENCE
10) Lasso Regression: Simple Definition - Statistics How To

Додаток 1

код програми

```
<!DOCTYPE html>

<html lang="en">

<head>
```

<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9
JvoRxT2MZw1T" crossorigin="anonymous">

<link rel="stylesheet" href="../static/css/my.css">

<meta charset="UTF-8">

<title>Welcome</title>

</head>

<body>

<div class="view" style="background-image: url('../static/musictuts.jpg');
background-repeat: no-repeat;

background-size: cover; background-position: center center;">

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
id=mybgr"style="background-color:

rgba(102, 3, 65, 0.4)">

<h1 class="display-4 font-weight-normal" style="color:white">Прогнозування
рейтингу фільмів</h1>

```
{% block content %}

{% endblock %}

</div>

</body>

</html>
<html lang="en">

<head>

<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9
JvoRxT2MZw1T" crossorigin="anonymous">

<link rel="stylesheet" href="../static/css/my.css">

<meta charset="UTF-8">

</head>

<body>


<div    class="view"  style="background-image:  url('../static/musictuts.jpg');
background-repeat: no-repeat;

background-size: cover; background-position: center center;">

<div class="position-relative overflow-hidden p-3 p-md-5 m-md-3 text-center
id=mybgr"style="background-color:

rgba(102, 3, 65, 0.4)">

<div class="col-md-5 p-lg-5 mx-auto my-5">
```

```
 <body>

 <form name="test" method="post" action="input1.php">
  <p><b>Жанр:</b><br>
   <input type="text" size="40">
   <p><b>Акторький склад:</b><br>
   <form>
   <p><input type="button" value="Додати"></p>
  </form>
   <p><b>Режисер:</b><br>
   <input type="text" size="40">
      <p><b>Рік випуску:</b><br>
   <input type="text" size="40">
   <form>
   <p><input type="button" value="Прогноз"></p>
  </form>




 </form>

 </body>
</html>

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

from imblearn.over_sampling import SMOTE

import matplotlib.pyplot as plt

from sklearn.pipeline import make_pipeline

from sklearn.ensemble import GradientBoostingRegressor
```

```
from sklearn.model_selection import KFold, cross_val_score, train_test_split

from sklearn.metrics import mean_squared_error, r2_score

data_new['Genre'] = data_new['Genre'].str.strip('[]').str.replace(' ','').str.replace("'","')

data_new['Genre'] = data_new['Genre'].str.split(',')

plt.subplots(figsize=(12,10))

list1 = []

for i in data_new['Genre']:

  list1.extend(i)

ax =
pd.Series(list1).value_counts()[:10].sort_values(ascending=True).plot.barh(width=0.9
)

for i, v in
enumerate(pd.Series(list1).value_counts()[:10].sort_values(ascending=True).values):

  ax.text(.8, i, v,fontsize=12,color='white',weight='bold')

plt.title('Top Genres')

plt.show()


genreList = []

for index, row in data_new.iterrows():

  genres = row["Genre"]


  for genre in genres:

        if genre not in genreList:

        genreList.append(genre)

print(genreList[:10])
```

```
def binary(genre_list):
  binaryList = []

  for genre in genreList:
        if genre in genre_list:
        binaryList.append(1)
        else:
        binaryList.append(0)

  return binaryList

data_new['genres_bin'] = data_new['Genre'].apply(lambda x: binary(x))
print(data_new['genres_bin'].head())
data_new['Actors'] = data_new['Actors'].str.strip('[]').str.replace(' ','').str.replace("'",'')
data_new['Actors'] = data_new['Actors'].str.split(',')
plt.subplots(figsize=(12,10))
list1=[]
for i in data_new['Actors']:
  list1.extend(i)
ax=pd.Series(list1).value_counts()[:15].sort_values(ascending=True).plot.barh(width
=0.9)
for i, v in
enumerate(pd.Series(list1).value_counts()[:15].sort_values(ascending=True).values):
  ax.text(.8, i, v,fontsize=10,color='white',weight='bold')
```

```
plt.title('Actors with highest appearance')
plt.show()
for i, j in zip(data_new['Actors'], data_new.index):
  list2 = []
  list2 = i[:4]
  data_new.loc[j, 'cast'] = str(list2)
for i, j in zip(data_new['Actors'], data_new.index):
  list2 = []
  list2 = i
  list2.sort()
  data_new.loc[j, 'Actors'] = str(list2)
castList = []
for index, row in data_new.iterrows():
  cast = row["Actors"]

  for i in cast:
        if i not in castList:
        castList.append(i)
data_new['cast_bin'] = data_new['Actors'].apply(lambda x: binary(x))
print(data_new['cast_bin'].head())

def xstr(s):
  if s is None:
        return ''
```

```
  return str(s)

data_new['Director'] = data_new['Director'].apply(xstr)

plt.subplots(figsize=(12,10))

ax =
data_new[data_new['Director']!=''].Director.value_counts()[:10].sort_values(ascendin
g=True).plot.barh(width=0.9)

for i, v in
enumerate(data_new[data_new['Director']!=''].Director.value_counts()[:10].sort_value
s(ascending=True).values):

  ax.text(.5, i, v,fontsize=12,color='white',weight='bold')

plt.title('Directors with highest rated movies')

plt.show()

directorList=[]

for i in data_new['Director']:

  if i not in directorList:

        directorList.append(i)

data_new['director_bin'] = data_new['Director'].apply(lambda x: binary(x))

print(data_new['director_bin'].head())

lasso = linear_model.Lasso(alpha=1, normalize=True)


lasso.fit(X_train, y_train)

coefs = lasso.coef_

intercept = lasso.intercept_


print("b_0 =", intercept)
```

```
pd.DataFrame(data=np.expand_dims(coefs, 0), columns=X_train.columns)

r2_train = lasso.score(X_train, y_train)

r2_valid = lasso.score(X_val, y_val)

print(r2_train, r2_valid)

alpha_arr = [10000, 1000, 300, 200, 100, 30, 10, 3, 1, 0.3, 0.1, 0.03, 0.01]

models = []

for alpha in alpha_arr:

lasso = linear_model.Lasso(alpha=alpha, normalize=True)

lasso.fit(X_train, y_train)

models.append(lasso)

r2 = lasso.score(X_val, y_val)


results = pd.DataFrame({'y_true': np.squeeze(Y_test), 'y_pred': Y_pred})

results.head(10)
```