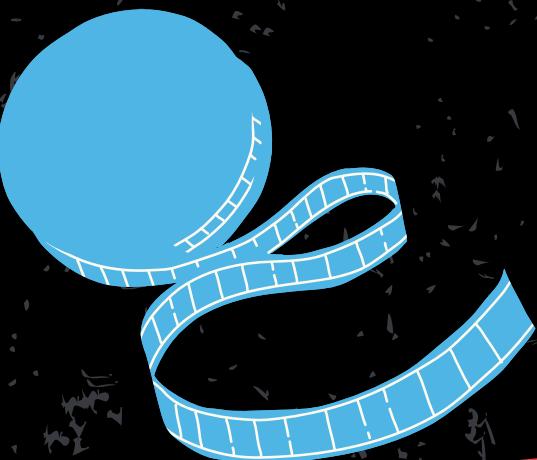


Project : Progress?

หัวข้อ: ภาพยันต์ร้อยอดนิยมที่ได้รับการโหวต



นำเสนอโดย



นรบดี สรีวิทูร 65102010195



ภาณุณัฐ พัฒนะจิราันันท 65102010200

วิชา CP 462 introduction to data science

ที่มาและความสำคัญ★

ในยุคดิจิทัลการจัดอันดับกาพยนตร์ยอดนิยมช่วยให้พูชมเลือก
กาพยนตร์ที่ตรงกับความสนใจได้ง่ายขึ้น การวิเคราะห์เรตติ้งจึง
สำคัญในการเข้าใจพฤติกรรมของพูชมและพัฒนาระบบแนะนำให้มี
ประสิทธิภาพ



ระบุปัญหาที่ต้องการ แก้ไขด้วยข้อมูล

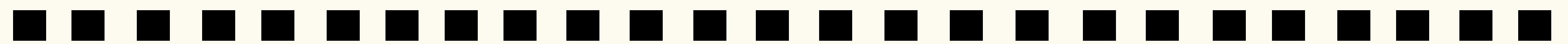
ผู้ชุมภาพยนตร์มักพบปัญหาในการค้นหาภาพยนตร์ที่ตรงกับความสูนใจ เนื่องจากข้อมูลมีจำนวนมากและไม่เป็นระบบ การจัดทำข้อมูล雷ตติ้งและการจัดอันดับที่ชัดเจนจะช่วยให้ผู้ชมเข้าถึงและเลือกชมภาพยนตร์ที่สอดคล้องกับรสชาติโดยอย่างสะดวกและรวดเร็วมากขึ้น



ระบุว่าการแก้ปัญหานี้จะ ★ เป็นประโยชน์กับใคร

การแก้ปัญหานี้จะเป็นประโยชน์ต่อพูชมในการเลือกภาพนตรที่ตรงกับความสนใจได้ง่ายยิ่งขึ้น นักวิเคราะห์สามารถใช้ข้อมูลเพื่อศึกษาความนิยมของภาพนตรอย่างมีประสิทธิภาพ นักพัฒนาสามารถสร้างระบบแนะนำที่แม่นยำยิ่งขึ้น และอุตสาหกรรมภาพนตรสามารถใช้ข้อมูลนี้เพื่อปรับกลยุทธ์ในการผลิตและจัดจำหน่ายตามแนวโน้มการตลาด





• **S:U TYPE OF LEARNING/TASK**



- 1. UNSUPERVISED LEARNING TASK: CLUSTERING**
- 2. REGRESSION TASK : LINEAR REGRESSION**
- 3. CONTENT-BASED FILTERING TASK:
RECOMMENDATION SYSTEM BASED ON CONTENT.**



Data acquisition ★

ชุดข้อมูลภาพยนตร์ที่ได้รับคะแนนสูงสุดตั้งแต่ปี 1874-2020 เป็นคอลเลกชันที่ครอบคลุม
เกือบทุกข้อมูลเกี่ยวกับภาพยนตร์ที่มีเรตติ้งสูง ซึ่งมาจากการแพลตฟอร์มการให้คะแนน
ภาพยนตร์ต่างๆ ชุดข้อมูลนี้มีแอ็ตกริบิวต์ที่หลากหลายสำหรับภาพยนตร์แต่ละเรื่อง เช่น
พู衡阳, คอลเลกชันที่เกี่ยวข้อง, งบประมาณ, ประเภทภาพยนตร์, โอมเพจ, rHss
ภาพยนตร์, rHssIMDb, ภาษาต้นฉบับ, ชื่อเรื่องต้นฉบับ, คำอธิบาย, ความนิยม, เส้นทาง
โปรดักชัน, บริษัทผู้ผลิต, ประเภทผู้ผลิต, วันที่ออกฉาย, รายได้, ระยะเวลาฉาย, ภาษาที่ใช้ใน
ภาพยนตร์, สถานะ, สโลแกน, ชื่อเรื่อง, วิดีโอ, คะแนนเฉลี่ย, จำนวนโหวต
ซึ่งมีประโยชน์อย่างยิ่งสำหรับการวิเคราะห์ข้อมูลเชิงสำรวจ การสร้างระบบคำแนะนำ และ
ทำความเข้าใจแนวโน้มของภาพยนตร์ซึ่งมี จำนวนข้อมูล 45466 ข้อมูลในแต่ละหลัก



1

Data manipulate/Cleansing

```
df.isnull().sum()

adult          0
belongs_to_collection    0
budget          0
genres          0
homepage         0
id              0
imdb_id         0
original_language    0
original_title      0
overview         0
popularity        0
poster_path        0
production_companies    0
production_countries     0
release_date       0
revenue           0
runtime           0
spoken_languages      0
status            0
tagline           0
title             0
video             0
vote_average       0
vote_count         0

dtype: int64

[115] #Convert date time
df['release_date'] = pd.to_datetime(df['release_date'], errors = 'coerce')

[116] df = df.dropna()

[117] data_cleaned_columns = df.dropna(axis=1)

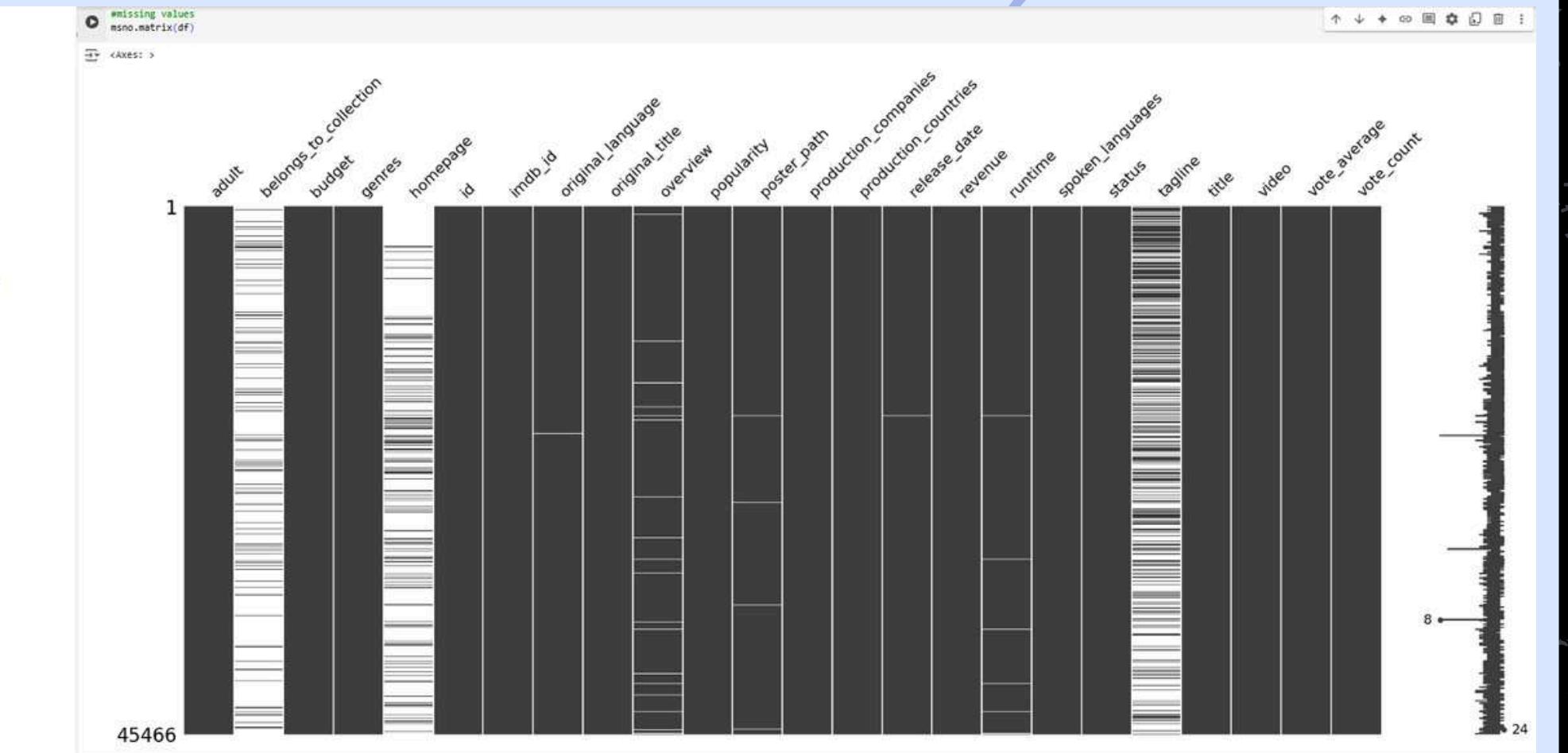
[118] df.duplicated().sum()

[119] df.drop_duplicates(inplace = True)

[120] df.columns
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'video',
       'vote_average', 'vote_count'],
      dtype='object')
```

1

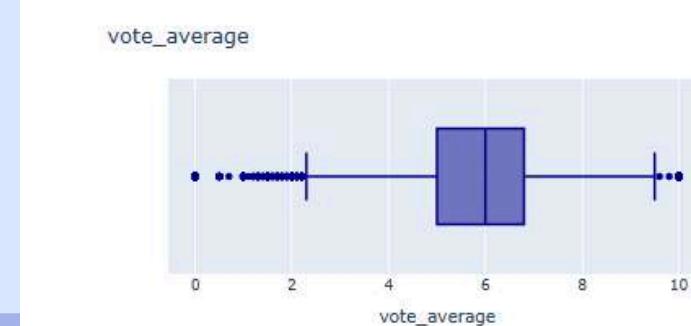
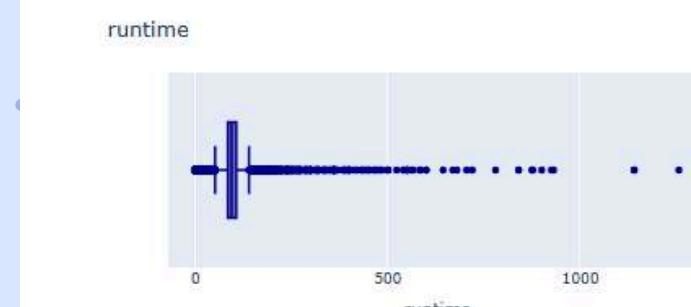
Data manipulate/Cleansing



1

Data manipulate/Cleansing

```
[33]: #outlier
for i in df.select_dtypes(include = 'number').columns:
    fig = px.box(df, x = i)
    fig.update_layout(
        title = f'{i}',
        height = 300,
        width = 600
    )
    fig.update_traces(marker_color='#01008c')
    fig.show()
```



2

Exploratory Data Analysis

Code K1Top 10 Movies with Highest vote count

```
[46] top_10_votes = df.nlargest(10, 'vote_count')
    fig = px.bar(top_10_votes,
                  x = 'title',
                  y = 'vote_count',
                  title = 'Top 10 Movies with Highest Vote Count')
    fig.update_traces(marker_color='#01008c')
    fig.show()
```

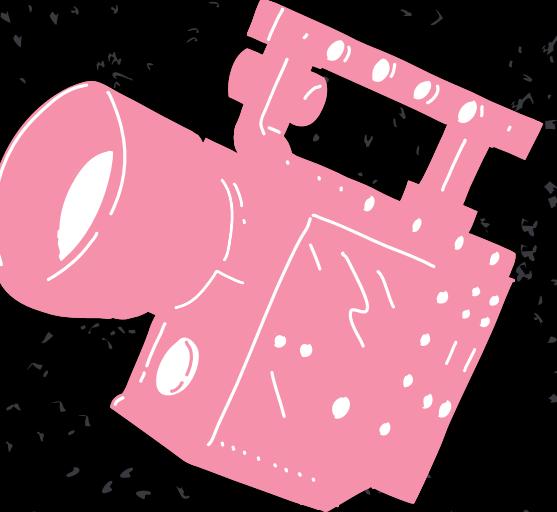
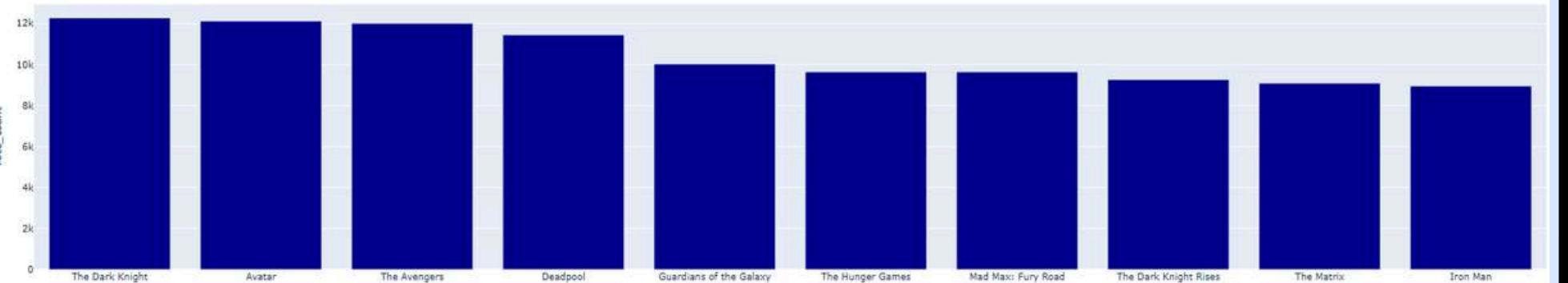
2



?

กราฟจาก Code ก่อนหน้านี้

Top 10 Movies with Highest Vote Count



?



2

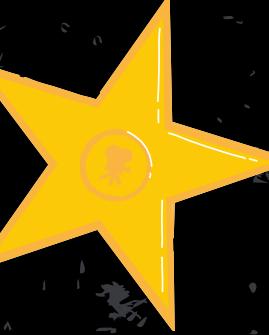
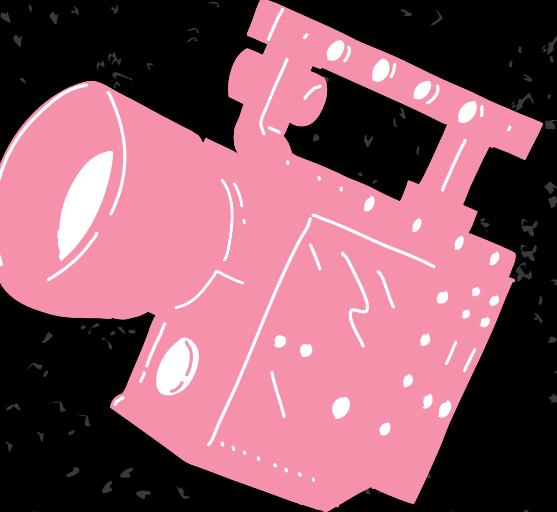
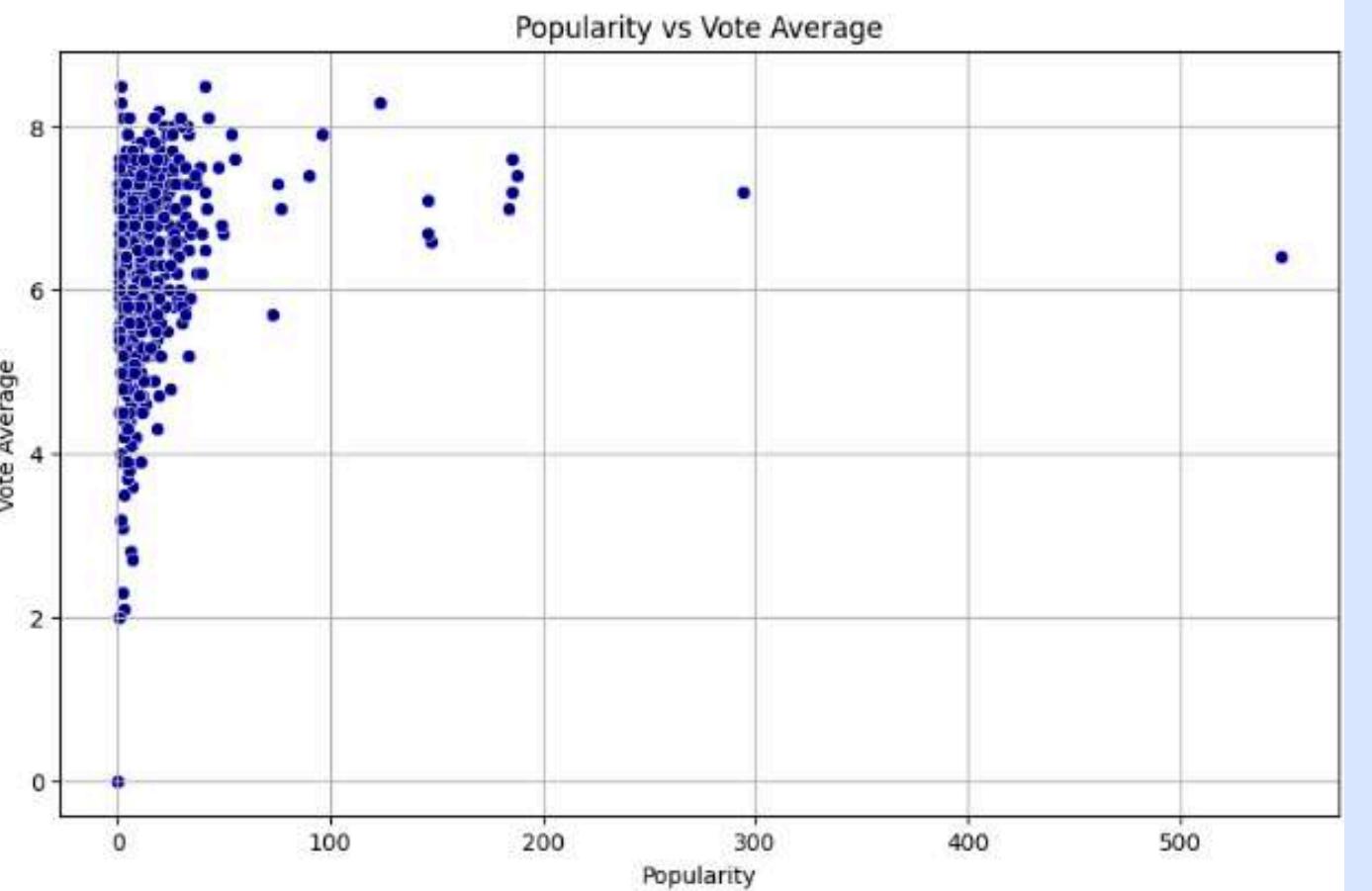
Code ห1 Popularity vs Vote average

```
[44] #Popularity vs Vote average
#ความนิยมนิยมต่อคะแนนโหวตเฉลี่ยหรือไม่
plt.figure(figsize=(10, 6))
sns.scatterplot(x='popularity', y='vote_average', data=df, color='#01008c')
plt.title('Popularity vs Vote Average')
plt.xlabel('Popularity')
plt.ylabel('Vote Average')
plt.grid()
plt.show()
```

2



กราฟจาก Code ก่อนหน้านี้



2

Code #1 Vote Count vs Vote average

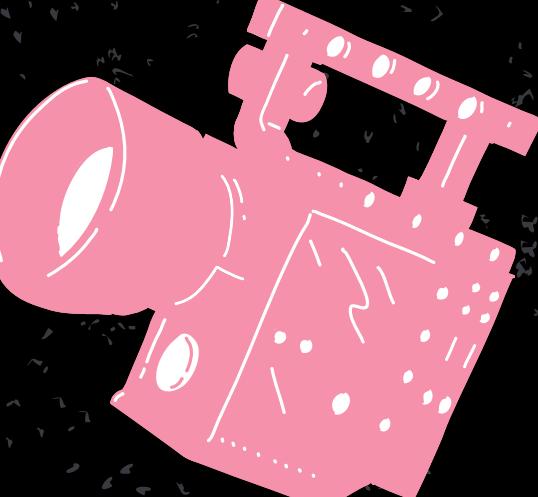
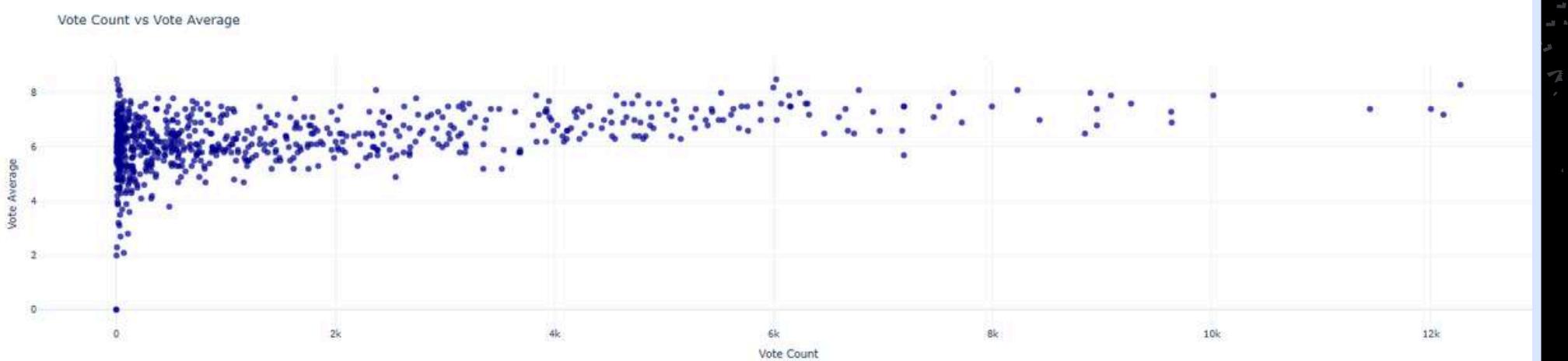
```
#Vote Count vs Vote average
#จำนวนการโหวต มีผลต่อ คะแนน平均เสียง
fig = px.scatter(df, x='vote_count', y='vote_average',
                  labels={'vote_count': 'Vote Count', 'vote_average': 'Vote Average'},
                  title='Vote Count vs Vote Average')
fig.update_traces(marker=dict(color='#01008c', size=8, opacity=0.7))
fig.update_layout(
    xaxis_title='Vote Count',
    yaxis_title='Vote Average',
    template='plotly_white'
)
fig.show()
```

2



?

กราฟจาก Code ก่อนหน้านี้



?



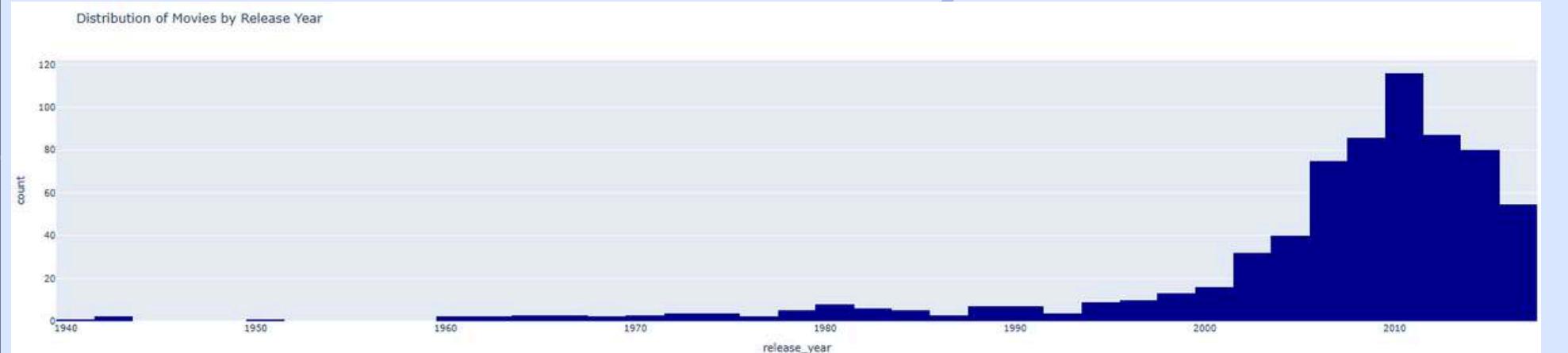
2

Code #1 Distribute of Movies by Release Year

```
[1]: # ក្រោមឯណ៍នេះការអំពេកភាពយនទរួចយ៉ាងវិនិត្តខ្សោយលោ។  
df['release_date'] = pd.to_datetime(df['release_date'])  
df['release_year'] = df['release_date'].dt.year  
fig = px.histogram(df, x='release_year', title='Distribution of Movies by Release Year')  
fig.update_traces(marker_color='#01008c')  
fig.show()
```

2

กราฟจาก Code ก่อนหน้านี้



?

2

Code H1 Vote Average by Release Year

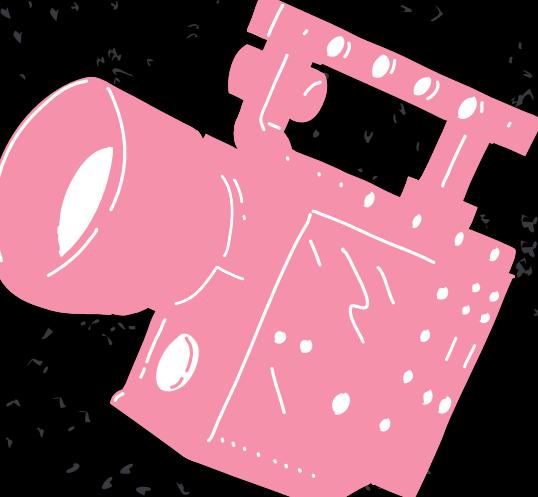
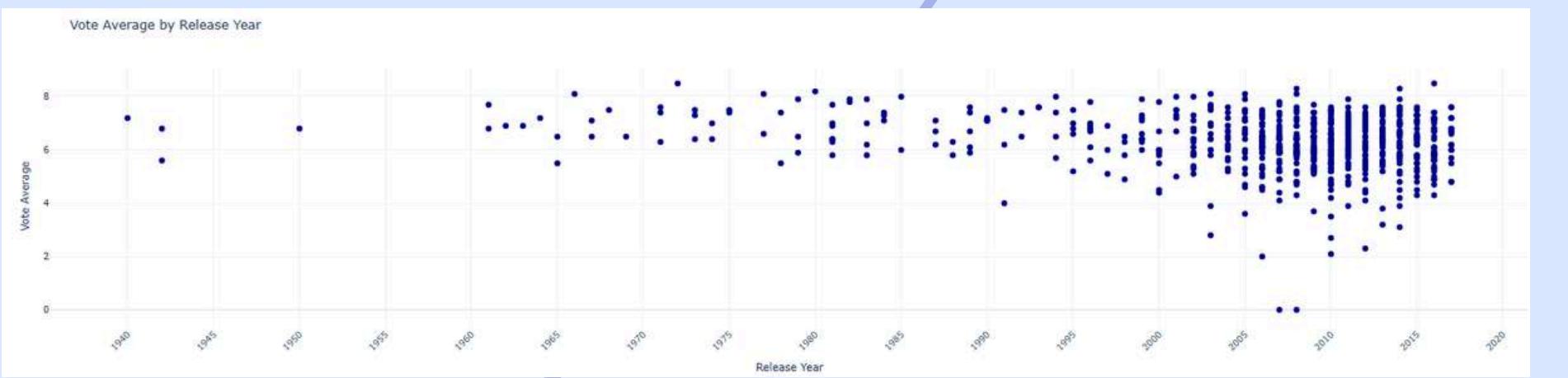
```
#គុណនាំថាគារលើខ្លួនរបស់ខ្លួនមិនមែនដែលត្រូវការងារទេ  
fig = px.scatter(df, x='release_year', y='vote_average',  
                  labels={'vote_average': 'Vote Average', 'release_year': 'Release Year'},  
                  title='Vote Average by Release Year')  
  
fig.update_traces(marker=dict(color='#01008c', size=8))  
  
fig.update_layout(  
    xaxis_title='Release Year',  
    yaxis_title='Vote Average',  
    template='plotly_white',  
    xaxis=dict(tickmode='linear', tick0=1950, dtick=5),  
    xaxis_tickangle=-45 )  
fig.show()
```

2

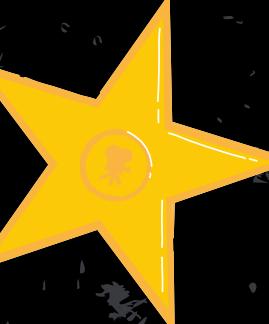


?

กราฟจาก Code ก่อนหน้านี้



?



2

Code ที่ 1 Movie Released per Year

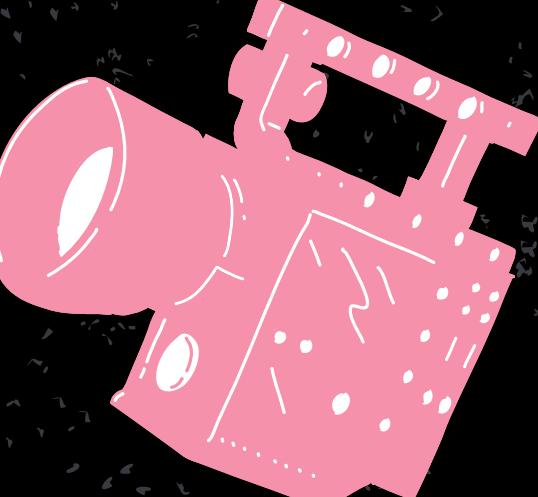
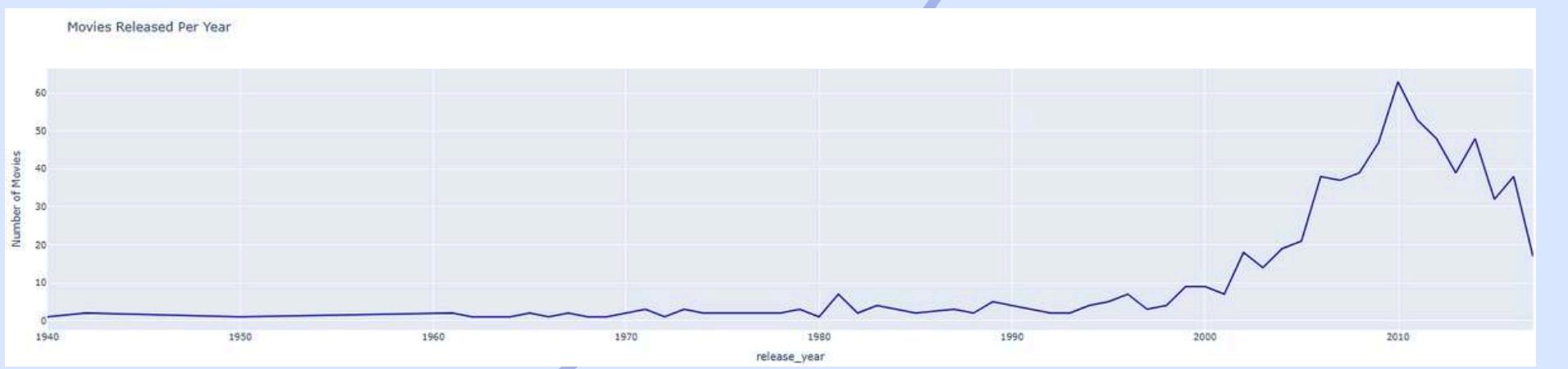
```
#เขียนโปรแกรมออกผลลัพธ์ของจำนวนหนังในแต่ละปี
movies_per_year = df['release_year'].value_counts().sort_index()
fig = px.line(movies_per_year,
               x = movies_per_year.index,
               y = movies_per_year.values,
               labels = {'x': 'Year', 'y': 'Number of Movies'},
               title = 'Movies Released Per Year')
fig.update_traces(line=dict(color='#01008c'))
fig.show()
```

2



?

กราฟจาก Code ก่อนหน้านี้



?



Summary



#ชื่อโครงการ: ก้าวยนต์ร์ก้า

#1. เป้าหมายของโครงการ

#ทำการวิเคราะห์เรตติ้งหนังก้าวยนต์ในแต่ละช่วงเวลาซึ่งจะทำให้ผู้ชมเลือกชมก้าวยนต์ที่สนุกและน่าสนใจได้ดียิ่งขึ้น และอัตสาหกรรมก้าวยนต์สามารถใช้ข้อมูลนี้เพื่อปรับกลยุทธ์ในการผลิตและจัดจำหน่าย

#สร้างโมเดลในการแนะนำหนังที่มีความคล้ายคลึงกับหนังที่เราเลือก

#2. ความถ้วนหน้าของโครงการจนถึงปัจจุบัน

#การกำกัลังส่องข้อมูล

#ทำการลูบข้อมูลที่ไม่สมบูรณ์ เช่น ข้อมูลที่มีค่าที่หายไปอย่างสำคัญและข้อมูลที่ไม่ถูกต้อง

#ข้อมูลที่มีค่า outlier ได้รับการตรวจสอบและแก้ไขตามความเหมาะสม

#การวิเคราะห์ข้อมูลเชิงสำรวจ

#ได้มีการทำกราฟที่มีแนวโน้มการผลิตก้าวยนต์อย่างไรในแต่ละช่วงเวลา

#กราฟแนวโน้มการอภิจัยของก้าวยนต์ในแต่ละปี

#กราฟคะแนนโหวตเฉลี่ยของก้าวยนต์เปลี่ยนแปลงไปตามปีที่อภิจัยหรือไม่

#กราฟความนิยมนิยมมีผลต่อคะแนนโหวตเฉลี่ยหรือไม่

#กราฟจำนวนการโหวต มีผลต่อคะแนนโหวตเฉลี่ย

#กราฟ 10 อันดับหนังที่ได้รับการโหวตมากที่สุด

#ความสำเร็จที่ได้

#ได้ข้อมูลเชิงลึกเกี่ยวกับหนังยอดนิยม

#สร้างความเข้าใจเกี่ยวกับปัจจัยที่มีผลต่อหนังก้าวยนต์ เช่น แนวหนังก้าวยนต์ที่คนดูรับชมมากที่สุดและคนโหวตมากที่สุด

#Import Library packages : pandas , missingno , matplotlib.pyplot , seaborn , numpy , plotly.express , warnings , stats , literal_eval , TfidfVectorizer , CountVectorizer , linear_kernel , cosine_similarity , SnowballStemmer , WordNetLemmatizer , wordnet , warnings



1

Data pre-processing

```
import pandas as pd
from sklearn.model_selection import train_test_split

# ด้วยการโหลดข้อมูล (สามารถปรับใช้กับข้อมูลของคุณ)
df = pd.read_csv('/content/drive/MyDrive/movies_metadata.csv', low_memory=False)

# สมมุติว่าเรามีไฟล์ X และ target (y) สำหรับการทำโมเดล
# เช่น X คือไฟล์ทั้งหมด (เช่น genres, title) และ y คือผลลัพธ์ (เช่น rating หรือ label)

# ด้วยการสร้างไฟล์ X และ target y
X = df[['genres']] # เช่น เลือกเฉพาะคอลัมน์ 'genres'
y = df['vote_average'] # หรือเลือกคอลัมน์ที่เป็น label เช่น 'vote_average'

# แบ่งข้อมูลเป็นชุดฝึกและชุดทดสอบ (test_size กำหนดตัวตัดส่วนของข้อมูลทดสอบ  เช่น 20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# แสดงข้อมูลที่แบ่งแล้ว
print(f"จำนวนข้อมูลทั้งหมด: {len(df)}")
print(f"จำนวนข้อมูลฝึก: {len(X_train)}")
print(f"จำนวนข้อมูลทดสอบ: {len(X_test)}")

# ด้วยการตรวจสอบข้อมูลบางส่วน
print("ข้อมูลฝึก (X_train):")
print(X_train.head())

print("ข้อมูลทดสอบ (X_test):")
print(X_test.head())
```

1

ผลลัพธ์จาก Code ก่อนหน้านี้

จำนวนข้อมูลทั้งหมด: 45466

จำนวนข้อมูลฝึก: 36372

จำนวนข้อมูลทดสอบ: 9094

ข้อมูลฝึก (X_train):

```
genres
41712 [{"id": 18, "name": "Drama"}, {"id": 53, "name...
25646 [{"id": 10751, "name": "Family"}, {"id": 35, "...
29596 [{"id": 36, "name": "History"}, {"id": 18, "na...
4526 [{"id": 18, "name": "Drama"}, {"id": 36, "name...
41710 [{"id": 10749, "name": "Romance"}, {"id": 35, ...
ข้อมูลทดสอบ (X_test):
```

```
genres
43526 [{"id": 18, "name": "Drama"}, {"id": 35, "name...
6383 [{"id": 18, "name": "Drama"}, {"id": 35, "name...
3154 [{"id": 80, "name": "Crime"}, {"id": 18, "name...
10146 [{"id": 27, "name": "Horror"}, {"id": 10749, "...
9531 [{"id": 35, "name": "Comedy"}, {"id": 18, "nam...
```

1

Build Machine Learning Model (Linear Regression)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import MultiLabelBinarizer

# อ่านข้อมูลจาก CSV
df = pd.read_csv('/content/drive/MyDrive/movies_metadata.csv', low_memory=False)

# ตัดแต่งภาษา genres
df['genres'] = df['genres'].apply(lambda x: eval(x) if isinstance(x, str) else [])
df['genre_names'] = df['genres'].apply(lambda x: [genre['name'] for genre in x])

# แปลงภาษา genres ด้วย One-Hot Encoding ใช้ MultiLabelBinarizer
mhb = MultiLabelBinarizer()
genres_onehot = mhb.fit_transform(df['genre_names'])

# สร้าง DataFrame ด้วย One-Hot Encoding
genres_df = pd.DataFrame(genres_onehot, columns=mhb.classes_)

# แทนที่ NaN ใน 'vote_average' ด้วย average
df['vote_average'] = df['vote_average'].fillna(df['vote_average'].mean())

# ตั้งค่า X และ y
X = genres_df # จะนำ genres ที่แปลงเป็น One-Hot Encoding
y = df['vote_average'] # ตั้งค่าเป้าหมาย (vote_average)

# แบ่งข้อมูลเป็นชุดฝึกหัดและทดสอบ
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

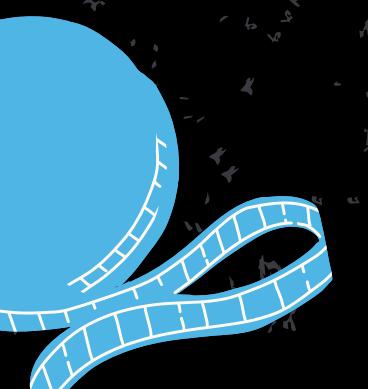
# สร้าง模型 Linear Regression
model = LinearRegression()

# ฝึก模型
model.fit(X_train, y_train)

# ทำนายผลลัพธ์
y_pred = model.predict(X_test)

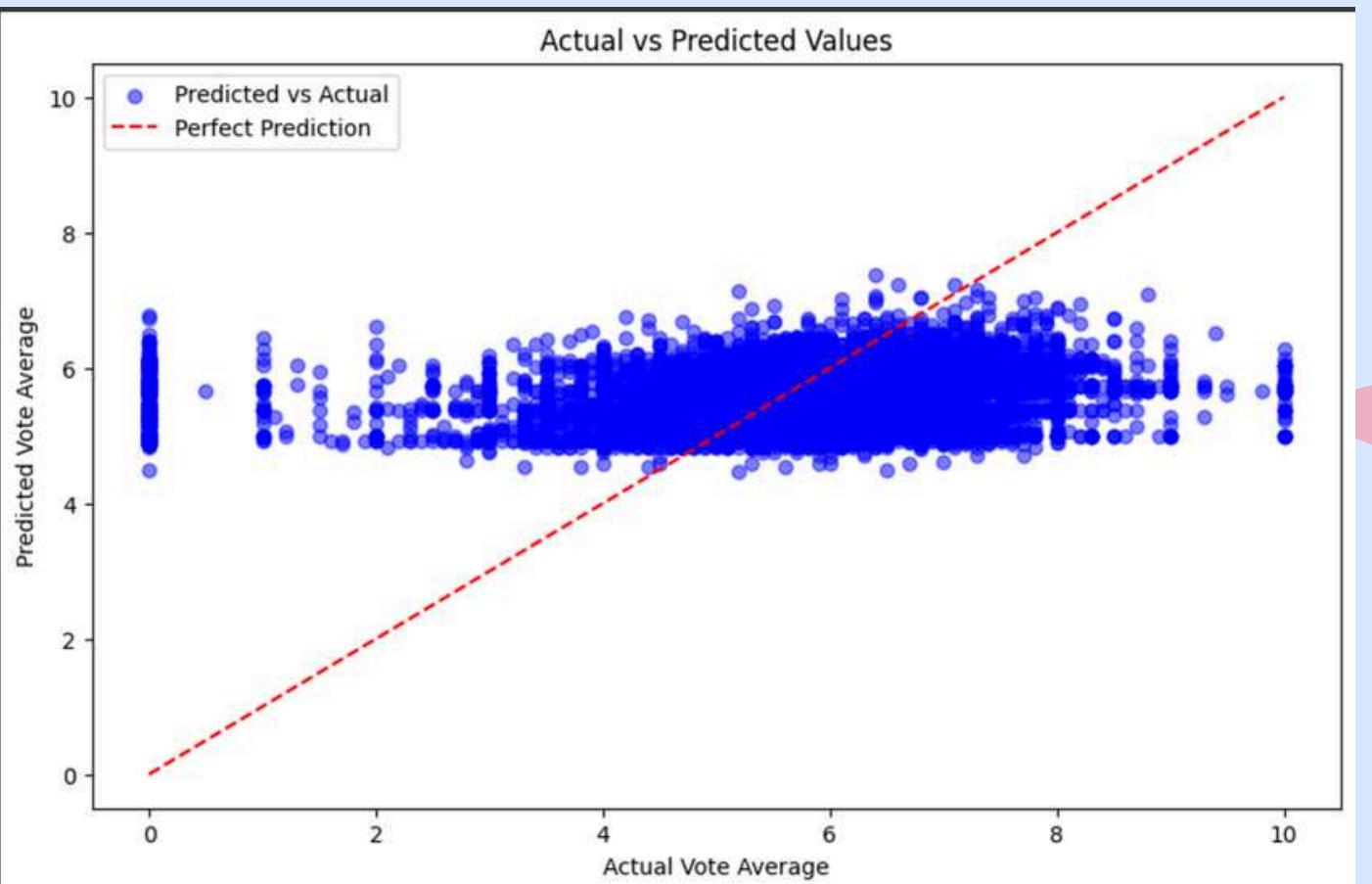
# คำนวณค่าความผิดพลาดต่างๆ
mae = mean_absolute_error(y_test, y_pred) # คือ MAE (Mean Absolute Error)
mse = mean_squared_error(y_test, y_pred) # คือ MSE (Mean Squared Error)
r2 = r2_score(y_test, y_pred) # คือ R^2 (Coefficient of Determination)

# แสดงผลลัพธ์
print("Mean Absolute Error: {mae}")
print("Mean Squared Error: {mse}")
print("R^2 Score: {r2}")
```



1

ผลลัพธ์จาก Code ก่อนหน้านี้



Mean Absolute Error: 1.3024770156602783
Mean Squared Error: 3.538108123292994
R² Score: 0.050750605296682294

1

Build Machine Learning Model (Content-Based Filtering)

```
# พิงค์ชั้นแนะนำหนัง
def recommend_movies(title, cosine_sim=cosine_sim):
    # หา index ของหนังที่เลือก
    idx = df[df['Title'] == title].index[0]

    # หาอันดับความคล้ายคลึง
    similarity_scores = list(enumerate(cosine_sim[idx]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

    # ติ๊ง index ของหนังที่คล้ายที่สุด (ยกเว้นตัวเอง)
    movie_indices = [i[0] for i in similarity_scores[1:4]] # แนะนำ 3 เรื่อง

    return df['Title'].iloc[movie_indices]

# ตัวอย่างการใช้งาน
movie_to_recommend = "Inception"
recommended_movies = recommend_movies(movie_to_recommend)

print(f"\nMovies recommended for '{movie_to_recommend}':")
print(recommended_movies.to_string(index=False))
```

```
DataFrame:
   Title           Genres
0  Inception  Sci-Fi, Action, Adventure
1  Interstellar  Sci-Fi, Drama, Adventure
2  The Matrix  Sci-Fi, Action, Thriller
3  The Martian  Sci-Fi, Adventure, Drama
4    Gravity  Sci-Fi, Drama, Thriller
```

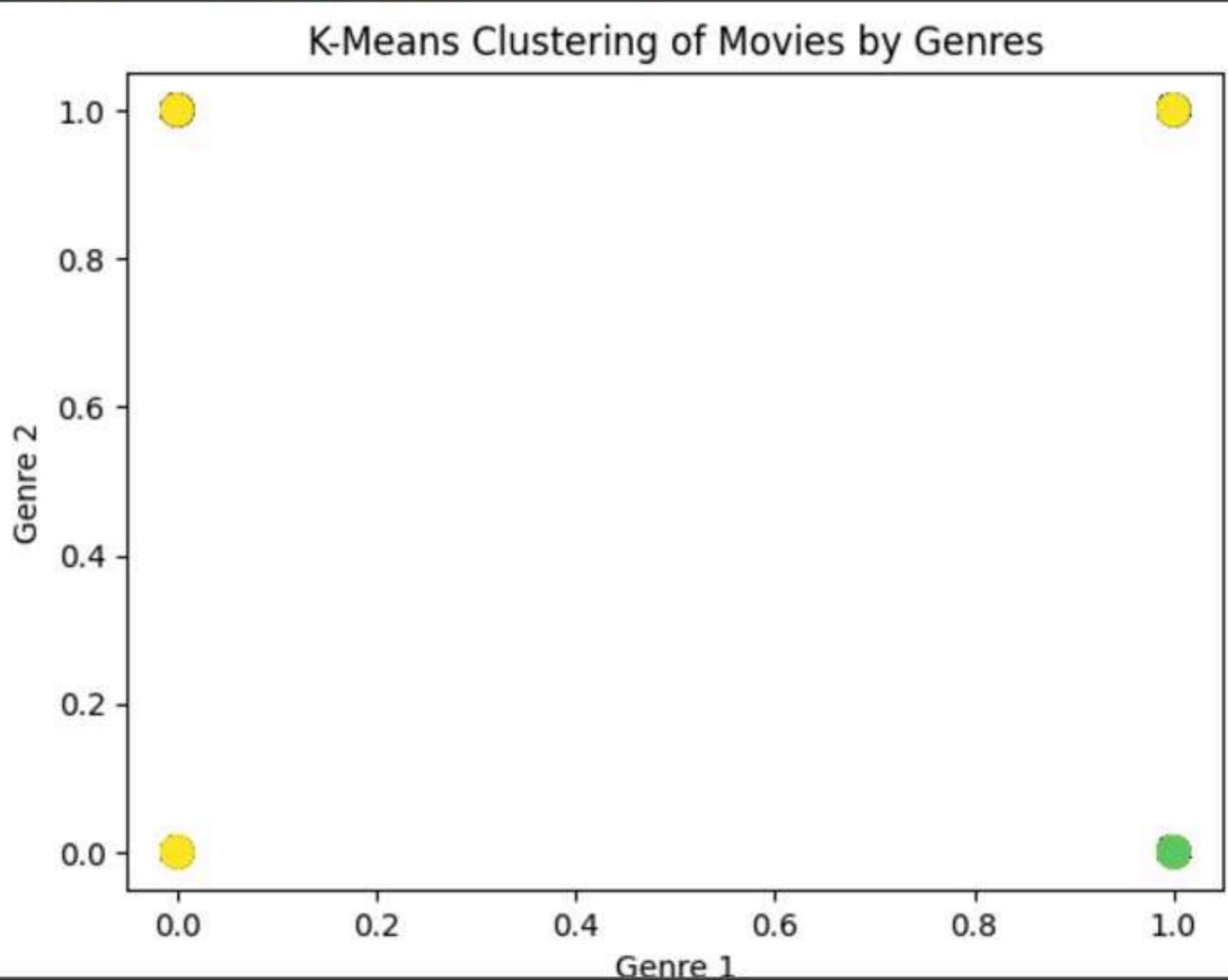
```
Movies recommended for 'Inception':
Interstellar
The Matrix
The Martian
```

1

ผลลัพธ์จาก Code ก่อนหน้านี้

Clustered Data by Genres:

	title	Cluster
0	Toy Story	0
1	Jumanji	0
2	Grumpier Old Men	2
3	Waiting to Exhale	2
4	Father of the Bride Part II	2



1

Build Machine Learning Model (Unsupervised Clustering)

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import MultiLabelBinarizer
import matplotlib.pyplot as plt

# อ่านข้อมูลจาก CSV
df = pd.read_csv('/content/drive/MyDrive/movies_metadata.csv')

# ตรวจสอบข้อมูล genres
df['genres'] = df['genres'].apply(lambda x: eval(x) if isinstance(x, str) else [])
df['genre_names'] = df['genres'].apply(lambda x: [genre['name'] for genre in x])

# แปลงข้อมูล genres เป็น One-Hot Encoding โดยใช้ MultiLabelBinarizer
mlb = MultiLabelBinarizer()
genres_onehot = mlb.fit_transform(df['genre_names'])

# สร้าง DataFrame สำหรับ One-Hot Encoding
genres_df = pd.DataFrame(genres_onehot, columns=mlb.classes_)

# การใช้ KMeans ในการจัดกลุ่มตาม genres
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(genres_df)

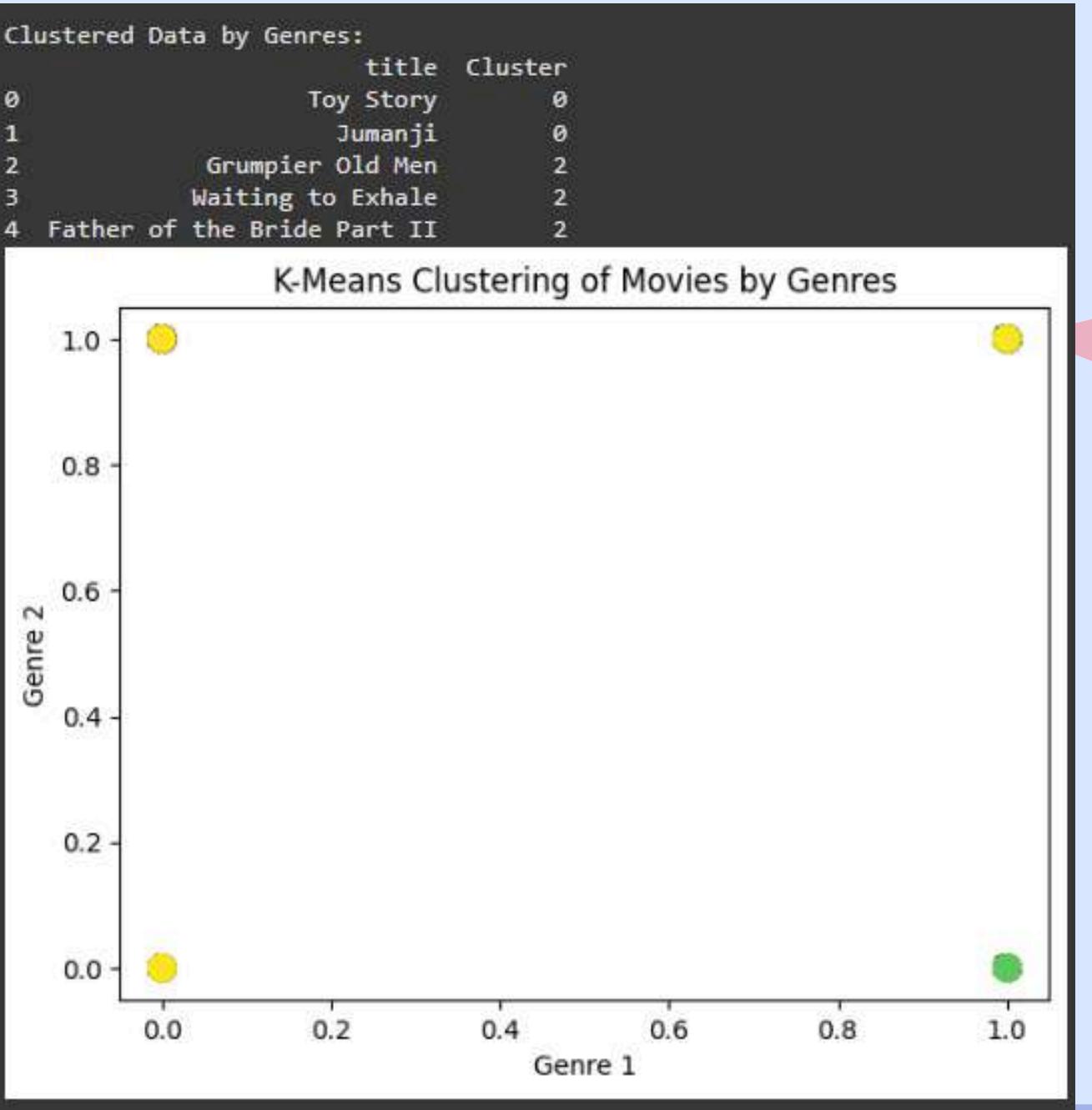
# เพิ่มคอลัมน์ Cluster ลงใน DataFrame
df['Cluster'] = kmeans.labels_

# แสดงข้อมูลที่ได้กลุ่มแล้ว
print("\nClustered Data by Genres:")
print(df[['title', 'Cluster']].head())

# แสดงกราฟ 2D ของ Cluster โดยใช้ 2 คอลัมน์แรกจาก genres_onehot
plt.scatter(genres_df.iloc[:, 0], genres_df.iloc[:, 1], c=df['Cluster'], cmap='viridis', s=100)
plt.title("K-Means Clustering of Movies by Genres")
plt.xlabel("Genre 1")
plt.ylabel("Genre 2")
plt.show()
```

1

ผลลัพธ์จาก Code ก่อนหน้านี้



1

Build Machine Learning Model (Evaluate Model)

MEAN ABSOLUTE ERROR: 1.3024770156602783

MEAN SQUARED ERROR: 3.538108123292994

R² SCORE: 0.050750605296682294

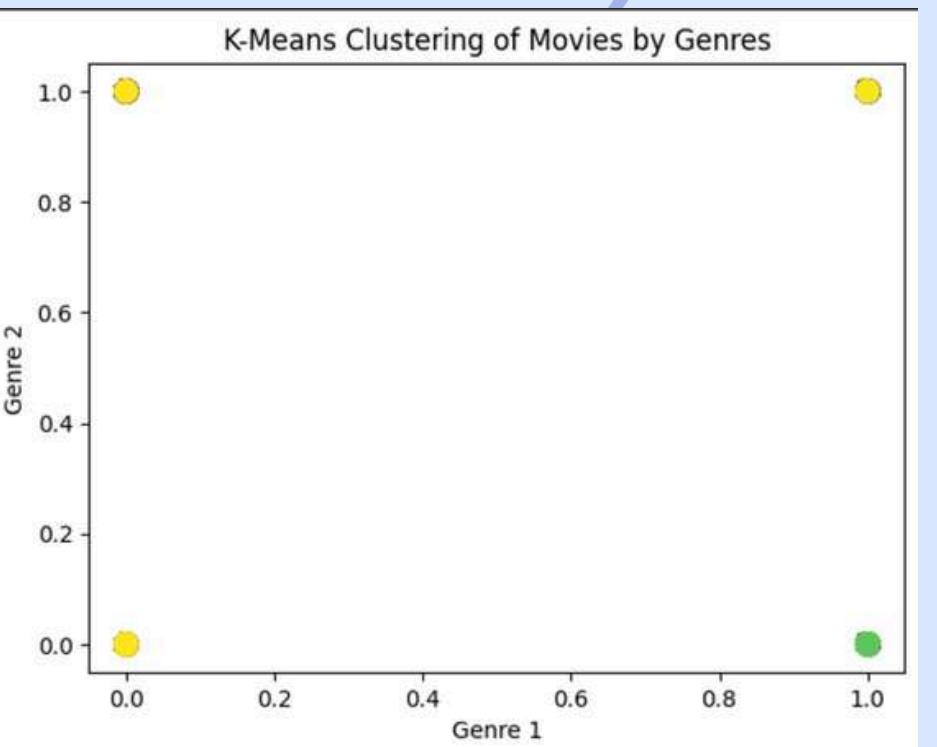
ค่าที่ได้จากโมเดล Linear Regression ที่ใช้ genres เป็นฟีเจอร์เพื่อ
คำนวณ vote_average ยังไม่สามารถคำนวณได้แม่นยำมากนัก:

MAE || MSE ค่อนข้างสูง || แสดงถึงความผิดพลาดในการคำนวณ
R² ก็ต่ำ (0.05) || แสดงว่าโมเดลสามารถอธิบายข้อมูลได้เพียงแค่ 5%
ซึ่งคงน่าจะต่ำ.

1

Build Machine Learning Model (Evaluate Model)

พลลัพท์ Unsupervised Clustering



สีของจุดคือจำแนกตามประเภทหนังที่คล้ายๆกัน ส่วนที่มันห่างกัน
เพราเว้มันคำนวนจากความคล้ายคลึงของประเภทหนังซึ่งแต่ละ
ประเภทมีความคล้ายคลึงกันแตกต่างกันอยูดีทำให้อยู่หาง
กัน

1

Build Machine Learning Model (Evaluate Model)

```
import pandas as pd
import json

# อ่านข้อมูลจาก CSV
df = pd.read_csv('/content/drive/MyDrive/movies_metadata.csv', low_memory=False)

# ฟังก์ชันเพื่อแปลงข้อมูล genres จาก JSON เป็น list ของแต่ละหนัง
def parse_genres(genres_str):
    try:
        genres = json.loads(genres_str)
        return [genre['name'] for genre in genres] # ถ้าเป็น array
    except (ValueError, TypeError):
        return []

# แปลง genres ทั้งหมดใน DataFrame
df['genres'] = df['genres'].apply(parse_genres)

# ฟังก์ชันตรวจสอบ genres ที่มีอยู่จริงใน list
def filter_genres(row):
    return isinstance(row, list) and len(row) > 0

# ฟังก์ชันคำนวณ Jaccard similarity ระหว่าง genres สองเรื่อง
def jaccard_similarity(genres1, genres2):
    # คำนวณ Jaccard similarity: จำนวนสิ่งที่สองเรื่องมีร่วม / จำนวนสิ่งที่สองเรื่องมี
    intersection = len(set(genres1).intersection(set(genres2)))
    union = len(set(genres1).union(set(genres2)))
    return intersection / union if union != 0 else 0

# ฟังก์ชันแนะนำหนังที่คล้ายกันเมื่อได้รับ input 'genres' พิมพ์บนหน้าจอ
def get_recommendations_genres(title):
    idx = df[df['title'] == title].index
    if len(idx) == 0:
        return f"'{title}' not found in the dataset."

    idx = idx[0] # ใช้ idx[0]
    target_genres = df.loc[idx]['genres']
    target_vote = df.loc[idx]['vote_average'] # ใช้คะแนน平均ของแต่ละเรื่อง

    # คำนวณความคล้ายคลึงกันของหนังที่อยู่ใน dataset โดยใช้ Jaccard similarity
    sim_scores = []
    for i, row in df.iterrows():
        if i != idx:
            sim_score = jaccard_similarity(target_genres, row['genres'])
            # รวมคะแนนโดยน้ำหนักของแต่ละหนัง
            weighted_score = sim_score * row['vote_average']
            sim_scores.append((i, weighted_score))

    # เรียงอัตโนมัติตามค่า score ที่ได้มาแล้ว
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    # 选取前 10 หนังที่คล้ายสุด (ถ้ามีมากกว่า)
    sim_scores = sim_scores[:10]

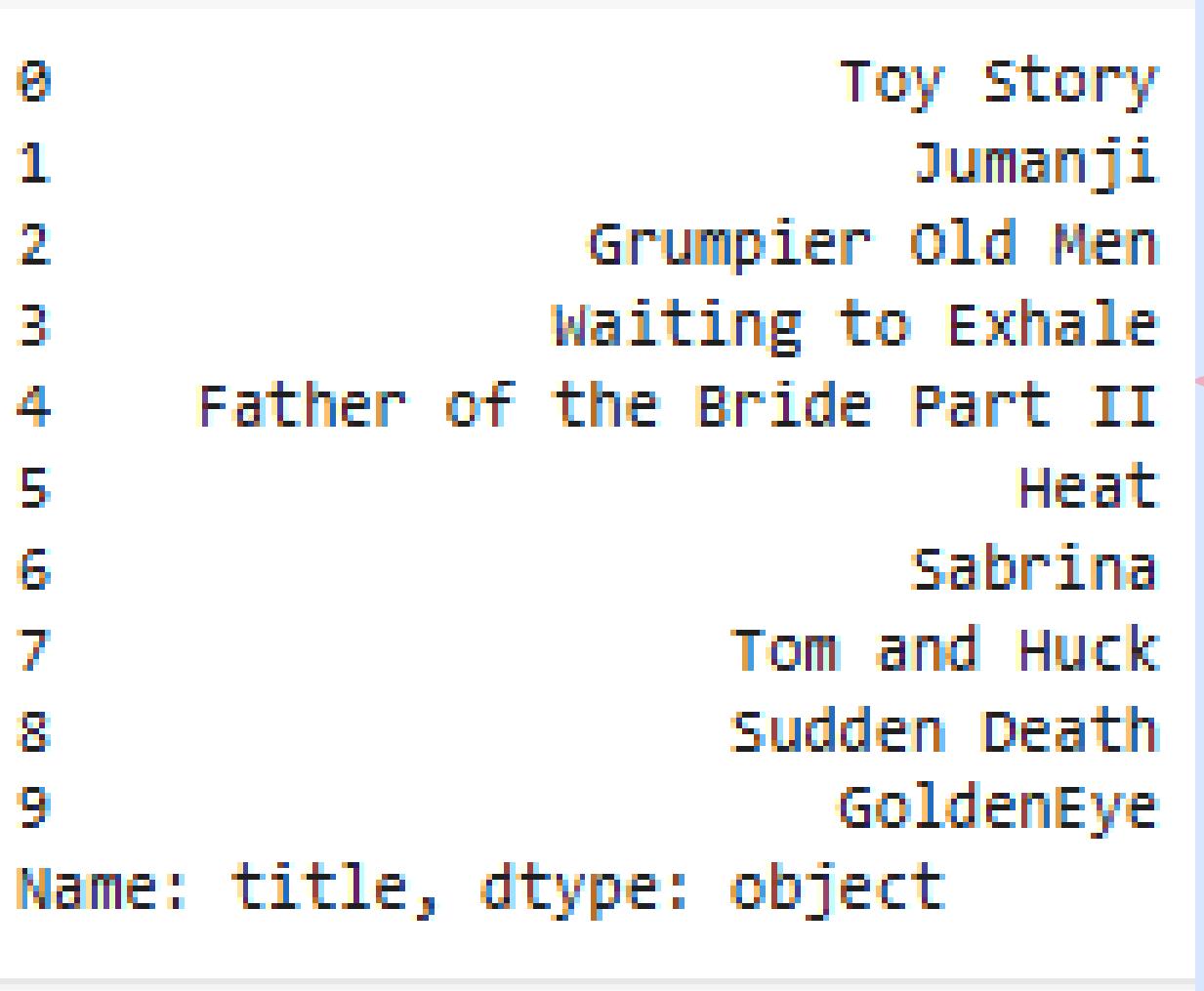
    # ดึงรหัสที่เก็บไว้
    movie_indices = [i[0] for i in sim_scores]
    return df['title'].iloc[movie_indices]

# ทดสอบการแนะนำ
recommended_movies = get_recommendations_genres('Deadpool')
print(recommended_movies)
```

?

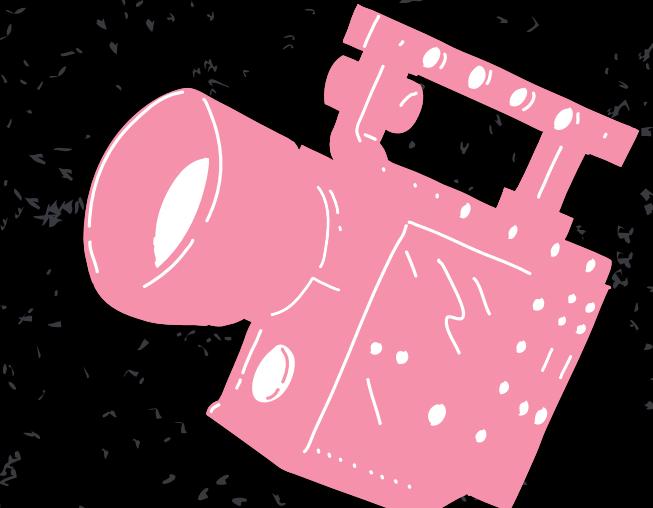
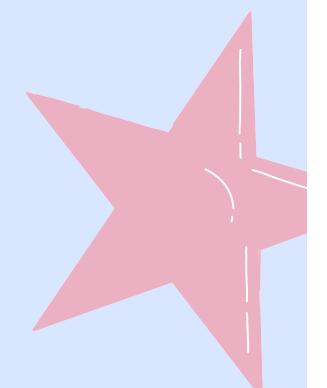
1

ผลลัพธ์จาก Code ก่อนหน้านี้



0 Toy Story
1 Jumanji
2 Grumpier Old Men
3 Waiting to Exhale
4 Father of the Bride Part II
5 Heat
6 Sabrina
7 Tom and Huck
8 Sudden Death
9 GoldenEye

Name: title, dtype: object



Summary



#1. สรุปการทำงานของโมเดล

#Linear Regression

#ค่าที่ได้แสดงว่าโมเดล Linear Regression ที่ใช้ genres เป็นฟีเจอร์เพื่อกำหนด vote_average ยังไม่สามารถกำหนดได้แม่นยำมากนัก:

#MAE และ MSE ค่อนข้างสูง แสดงถึงความผิดพลาดในการกำหนด

R^2 ที่ต่ำ (0.05) แสดงว่าโมเดลสามารถอธิบายข้อมูลได้เพียงแค่ 5% ซึ่งค่อนข้างต่ำ.

#Unsupervised Clustering จัดกลุ่มภาพยนตร์ตามประเภท (genres) โดยจะหากลุ่มที่มีลักษณะคล้ายกันตามประเภทของภาพยนตร์

#Content-Based Filtering เมื่อผู้ใช้ป้อนชื่อภาพยนตร์ เช่น "Inception" จะคำนวณ Cosine Similarity ระหว่างประเภทของภาพยนตร์ที่มีหัวใจเดียวกัน แล้วแนะนำ 3 ภาพยนตร์ที่มีความคล้ายคลึงสูงที่สุดตามประเภทภาพยนตร์เอง

#2. สรุปถึงการแก้ปัญหาที่ได้ตั้งไว้

#โมเดลได้ช่วยแนะนำประเภทหนังที่ตรงกับสนใจที่เราชอบดูหรือที่เราเลือกมาซึ่งจะช่วยให้ผู้ชมเข้าถึงและเลือกชมภาพยนตร์ที่สอดคล้องกับสนใจได้อย่างสะดวกและรวดเร็วมากขึ้น และอัตสาหกรรมภาพยนตร์สามารถใช้ข้อมูลนี้เพื่อปรับกลยุทธ์ในการผลิตและจัดจำหน่ายตามความต้องการของผู้ชม



Thank
you!