STC TV - كود تحليل سلوك المستخدم

الكود البرمجي المستخرج من Jupyter Notebook

تحليل فئات المشاهدين وأنماط المشاهدة بين SD و HD

1. إعداد البيئة وربط Google Drive

> هذا القسم يتضمن ربط Google Drive بـ Colab وتحديد مجلد العمل الأساسي.

```
import os

Root = "/content/drive/MyDrive/Colab Notebooks"

os.chdir(Root)
```

```
from google.colab import drive

drive.mount('/content/drive')
```

2. تثبيت المكتبات المطلوبة

> تثبيت مكتبة pyxlsb اللازمة لقراءة ملفات Excel بصيغة xlsb.

```
"""
تثبيت المكتبات اللازمة التي لم يتم تثبيتها افتراضيًا
مثال: pyxlsb
يمكنك إضافة أي مكتبة تخطط لاستخدامها.
"""
!pip install pyxlsb
```

3. استيراد المكتبات

> استيراد جميع المكتبات اللازمة لتحليل البيانات وإنشاء الرسوم البيانية.

```
# استيراد المكتبات المطلوبة
"""
يرجى استيراد أي مكتبات مطلوبة حسب احتياجاتك
"""
import pandas as pd      # يوفر هياكل عالية الأداء وسهلة الاستخدام وأدوات تحليل البيانات
import pyxlsb            # امتداد Excel لقراءة ملفات xlsb (ملف الإدخال)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import FuncFormatter
import os   # يوفر حسابات رياضية سريعة على المصفوفات والمصفوفات
```

4. قراءة البيانات والاستكشاف الأولي

> قراءة ملف البيانات وعرض المعلومات الأساسية عن هيكل البيانات.

```
# قراءة مجموعة بيانات Jawwy

dataframe = pd.read_excel("stc TV Data Set_T1.xlsb",
sheet_name="Final_Dataset")

# يرجى عمل نسخة من مجموعة البيانات إذا كنت ستعمل مباشرة وتجري تغييرات على مجموعة البيانات

#df=dataframe.copy()
```

```
# التحقق من شكل البيانات

dataframe.shape
```

```
# عرض أول 5 صفوف من البيانات

dataframe.head()
```

5. معالجة البيانات الأولية

> تنظيف البيانات وتوحيد قيم الأعمدة وتحويل أنواع البيانات.

```
# معالجة البيانات الأولية

print("جاري معالجة البيانات...")

if 'Column1' in dataframe.columns:

    dataframe = dataframe.drop(columns=['Column1'])


dataframe['program_name'] = dataframe['program_name'].str.strip()
```

```
# توحيد قيم عمود program_class

print("جاري توحيد قيم عمود program_class...")

# تحويل SERIES/EPISODES إلى SERIES

dataframe['program_class'] = dataframe['program_class'].replace('SERIES/
EPISODES', 'SERIES')

print(f"القيم الفريدة في عمود program_class بعد التوحيد:
{dataframe['program_class'].unique()}")
```

```
# طباعة معلومات عن البيانات

print("معلومات عن البيانات:")

print(f"أعمدة البيانات: {dataframe.columns.tolist()}")

print(f"أنواع البيانات: {dataframe.dtypes}")

print(f"عدد القيم المفقودة: {dataframe.isnull().sum().sum()}")
```

```
# تحويل الأعمدة الرقمية

dataframe[['duration_seconds', 'season', 'episode', 'series_title', 'hd']]
= dataframe[['duration_seconds', 'season', 'episode', 'series_title',
'hd']].apply(pd.to_numeric)

dataframe[['user_id_maped', 'program_name', 'program_class', 'program_desc',
'program_genre', 'original_name']] = dataframe[['user_id_maped',
'program_name', 'program_class', 'program_desc', 'program_genre',
'original_name']].astype(str)
```

6. الجزء الأول: تحليل فئات المشاهدين حسب صنف البرنامج

> تحليل توزيع المستخدمين وتصنيفهم حسب تفضيلاتهم للأفلام والمسلسلات.

```
# الجزء الأول: تحليل وتصنيف فئات المشاهدين وفقًا لصنف البرنامج

print("\n=== الجزء الأول: تحليل وتصنيف فئات المشاهدين وفقًا لصنف البرنامج
===\n")



# تحليل توزيع المستخدمين حسب صنف البرنامج

print("جاري تحليل توزيع المستخدمين حسب صنف البرنامج...")
```

```
# عدد المستخدمين الفريدين

unique_users = dataframe['user_id_maped'].nunique()

print(f"إجمالي عدد المستخدمين الفريدين: {unique_users}")



# عدد المستخدمين الذين يشاهدون الأفلام

movie_users = set(dataframe[dataframe['program_class'] == 'MOVIE']
['user_id_maped'].unique())

print(f"عدد المستخدمين الذين يشاهدون الأفلام: {len(movie_users)}")



# عدد المستخدمين الذين يشاهدون المسلسلات

series_users = set(dataframe[dataframe['program_class'] == 'SERIES']
['user_id_maped'].unique())

print(f"عدد المستخدمين الذين يشاهدون المسلسلات: {len(series_users)}")
```

```
# عدد المستخدمين الذين يشاهدون الأفلام فقط
movie_only_users = len(movie_users - series_users)
print(f"عدد المستخدمين الذين يشاهدون الأفلام فقط: {movie_only_users}")


# عدد المستخدمين الذين يشاهدون المسلسلات فقط
series_only_users = len(series_users - movie_users)
print(f"عدد المستخدمين الذين يشاهدون المسلسلات فقط: {series_only_users}")


# عدد المستخدمين الذين يشاهدون كلا النوعين
both_users = len(movie_users.intersection(series_users))
print(f"عدد المستخدمين الذين يشاهدون كلا النوعين: {both_users}")
```

7. تحليل وقت المشاهدة

> تحليل إجمالي ومتوسط وقت المشاهدة لكل صنف من البرامج.

```
# تحليل وقت المشاهدة حسب صنف البرنامج

print("\nجاري تحليل وقت المشاهدة حسب صنف البرنامج...")



# إجمالي وقت المشاهدة لكل صنف

total_duration_by_class = dataframe.groupby('program_class')
['duration_seconds'].sum()

print(f"إجمالي وقت المشاهدة حسب صنف البرنامج (بالثواني):
\n{total_duration_by_class}")



# متوسط وقت المشاهدة لكل صنف

avg_duration_by_class = dataframe.groupby('program_class')
['duration_seconds'].mean()

print(f"متوسط وقت المشاهدة حسب صنف البرنامج (بالثواني):
\n{avg_duration_by_class}")
```

8. تحليل تفضيلات الأنواع

> تحليل الأنواع الأكثر مشاهدة للأفلام والمسلسلات.

```
# تحليل تفضيلات الأنواع حسب صنف البرنامج

print("\nجاري تحليل تفضيلات الأنواع حسب صنف البرنامج...")



# الأنواع الأكثر مشاهدة للأفلام

movie_genres = dataframe[dataframe['program_class'] ==
'MOVIE'].groupby('program_genre').size().sort_values(ascending=False)

print(f"الأنواع الأكثر مشاهدة للأفلام:\n{movie_genres.head(10)}")



# الأنواع الأكثر مشاهدة للمسلسلات

series_genres = dataframe[dataframe['program_class'] ==
'SERIES'].groupby('program_genre').size().sort_values(ascending=False)

print(f"الأنواع الأكثر مشاهدة للمسلسلات:\n{series_genres.head(10) if not
series_genres.empty else 'لا توجد بيانات كافية للمسلسلات'}")
```

9. الجزء الثاني: تحليل أنماط المشاهدة حسب الجودة

> دراسة أنماط المشاهدة بين الجودة القياسية (SD) والجودة العالية (HD).

```
# الجزء الثاني: دراسة أنماط المشاهدة المختلفة للمستخدمين بين الجودة القياسية (SD) والجودة العالية (HD)

print("\n=== الجزء الثاني: دراسة أنماط المشاهدة المختلفة للمستخدمين بين الجودة القياسية (SD) والجودة العالية (HD) ===\n")



# تحليل أنماط المشاهدة حسب جودة العرض (SD و HD)

print("جاري تحليل أنماط المشاهدة حسب جودة العرض...")
```

```
# عدد المشاهدات بالجودة القياسية (SD) والجودة العالية (HD)

hd_counts = dataframe['hd'].value_counts()

print(f"عدد المشاهدات حسب جودة العرض:\n{hd_counts}")



# نسبة المشاهدات بالجودة القياسية (SD) والجودة العالية (HD)

hd_percentage = dataframe['hd'].value_counts(normalize=True) * 100

print(f"نسبة المشاهدات حسب جودة العرض:\n{hd_percentage}")
```

```
# تحليل جودة العرض حسب صنف البرنامج

hd_by_program_class = pd.crosstab(dataframe['program_class'],
dataframe['hd'])

print(f"عدد المشاهدات حسب صنف البرنامج وجودة العرض:
\n{hd_by_program_class}")



# نسبة جودة العرض حسب صنف البرنامج

hd_by_program_class_percentage = pd.crosstab(dataframe['program_class'],
dataframe['hd'], normalize='index') * 100

print(f"نسبة المشاهدات حسب صنف البرنامج وجودة العرض (%):
\n{hd_by_program_class_percentage}")
```

```
# تحليل جودة العرض حسب نوع البرنامج

hd_by_genre = pd.crosstab(dataframe['program_genre'],
dataframe['hd']).sort_values(by=1, ascending=False)

print(f"عدد المشاهدات حسب نوع البرنامج وجودة العرض (أعلى 10 أنواع بالجودة
العالية):\n{hd_by_genre.head(10)}")
```

10. إنشاء الرسوم البيانية

> إنشاء مجموعة من الرسوم البيانية لتوضيح النتائج والاتجاهات في البيانات.

```
# إنشاء الرسوم البيانية

print("\nجاري إنشاء الرسوم البيانية...")


# 1. رسم بياني لتوزيع المستخدمين حسب صنف البرنامج

plt.figure(figsize=(10, 6))

user_distribution = [movie_only_users, series_only_users, both_users]

labels = ['أفلام فقط', 'مسلسلات فقط', 'كلا النوعين']


# التحقق من وجود قيم غير صفرية قبل رسم الرسم البياني

if sum(user_distribution) > 0:

    plt.pie(user_distribution, labels=labels, autopct='%1.1f%%',
startangle=90, colors=['#ff9999','#66b3ff','#99ff99'])

    plt.title('توزيع المستخدمين حسب تفضيلات صنف البرنامج')

    plt.axis('equal')

    plt.savefig('user_distribution_by_program_class.png', dpi=300,
bbox_inches='tight')

else:

    print("تخطي رسم توزيع المستخدمين لعدم وجود بيانات كافية")

plt.close()
```

```
# 2. رسم بياني لإجمالي وقت المشاهدة حسب صنف البرنامج

plt.figure(figsize=(10, 6))

total_duration_by_class.plot(kind='bar', color=['#ff9999', '#66b3ff'])

plt.title('إجمالي وقت المشاهدة حسب صنف البرنامج')

plt.xlabel('صنف البرنامج')

plt.ylabel('إجمالي وقت المشاهدة (بالثواني)')

plt.xticks(rotation=0)

plt.savefig('total_duration_by_program_class.png', dpi=300,
bbox_inches='tight')

plt.close()
```

```
# 3. رسم بياني للأنواع الأكثر مشاهدة للأفلام

plt.figure(figsize=(12, 6))

if not movie_genres.empty:

    movie_genres.head(10).plot(kind='bar', color='#ff9999')

    plt.title('الأنواع الأكثر مشاهدة للأفلام')

    plt.xlabel('نوع الفيلم')

    plt.ylabel('عدد المشاهدات')

    plt.xticks(rotation=45, ha='right')

    plt.tight_layout()

    plt.savefig('top_movie_genres.png', dpi=300, bbox_inches='tight')

else:

    print("تخطي رسم الأنواع الأكثر مشاهدة للأفلام لعدم وجود بيانات كافية")

plt.close()
```

11. المزيد من الرسوم البيانية

> رسوم بيانية إضافية لتحليل المسلسلات وجودة العرض

```
# 4. رسم بياني للأنواع الأكثر مشاهدة للمسلسلات
plt.figure(figsize=(12, 6))
if not series_genres.empty and len(series_genres) > 0:
    series_genres.head(10).plot(kind='bar', color='#66b3ff')
    plt.title('الأنواع الأكثر مشاهدة للمسلسلات')
    plt.xlabel('نوع المسلسل')
    plt.ylabel('عدد المشاهدات')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.savefig('top_series_genres.png', dpi=300, bbox_inches='tight')
else:
    print("تخطي رسم الأنواع الأكثر مشاهدة للمسلسلات لعدم وجود بيانات كافية")
plt.close()
```

```
# 5. رسم بياني لنسبة المشاهدات حسب جودة العرض

plt.figure(figsize=(8, 6))

hd_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90,
colors=['#ff9999','#66b3ff'])

plt.title('نسبة المشاهدات حسب جودة العرض')

plt.ylabel('')

plt.legend(['SD', 'HD'])

plt.axis('equal')

plt.savefig('viewing_quality_distribution.png', dpi=300,
bbox_inches='tight')

plt.close()
```

```
# 6. رسم بياني لنسبة جودة العرض حسب صنف البرنامج

plt.figure(figsize=(10, 6))

hd_by_program_class_percentage.plot(kind='bar', stacked=True,
color=['#ff9999','#66b3ff'])

plt.title('نسبة جودة العرض حسب صنف البرنامج')

plt.xlabel('صنف البرنامج')

plt.ylabel('النسبة المئوية')

plt.xticks(rotation=0)

plt.legend(['SD', 'HD'])

plt.savefig('quality_by_program_class.png', dpi=300, bbox_inches='tight')

plt.close()
```

```
# 7. رسم بياني للأنواع الأكثر مشاهدة بالجودة العالية
plt.figure(figsize=(12, 6))
if 1 in hd_by_genre.columns and not hd_by_genre.empty:
    hd_by_genre.head(10)[1].plot(kind='bar', color='#66b3ff')
    plt.title('الأنواع الأكثر مشاهدة بالجودة العالية (HD)')
    plt.xlabel('نوع البرنامج')
    plt.ylabel('عدد المشاهدات')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.savefig('top_hd_genres.png', dpi=300, bbox_inches='tight')
else:
    print("تخطي رسم الأنواع الأكثر مشاهدة بالجودة العالية لعدم وجود بيانات كافية")
plt.close()
```

12. إنهاء التحليل

> طباعة رسالة إنهاء التحليل وتأكيد اكتمال جميع الخطوات.

```
# كتابة تقرير التحليل
print("\nجاري كتابة تقرير التحليل...")


# طباعة التقرير في co-lab
print("\n" + "="*50 + "\n")
print("تم الانتهاء من التحليل وكتابة التقرير بنجاح!")
print("\n" + "="*50 + "\n")
```

ملاحظات مهمة

> **تعليمات للاستخدام:**
> 1. تأكد من رفع ملف البيانات "stc TV Data Set_T1.xlsb" إلى مجلد Colab
> 2. قم بتشغيل الخلايا بالترتيب المعروض
> 3. ستحصل على تحليل شامل لسلوك المستخدمين ورسوم بيانية توضيحية
> 4. يمكن تعديل المسارات والمتغيرات حسب احتياجاتك
>
> **المخرجات المتوقعة:**
> - تحليل تفصيلي لفئات المشاهدين
> - رسوم بيانية لتوزيع المستخدمين
> - تحليل أنماط المشاهدة حسب الجودة
> - إحصائيات شاملة عن سلوك المستخدمين