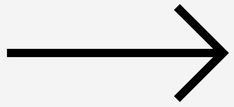
DSCI_633-Foundations Of Data Science & Analytics Project

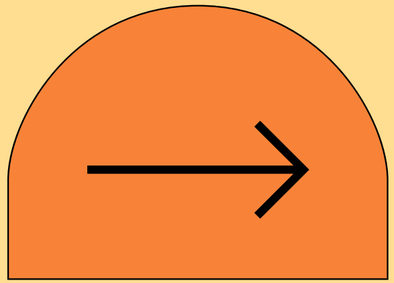
Hotstar Predict The Segment

Submitted by :
Team 3
Omkar Khanvilkar
Pranav Nair
Sujan Dutta
Varun Tandon



Overview

HOTSTAR USER SENTIMENT ANALYSIS

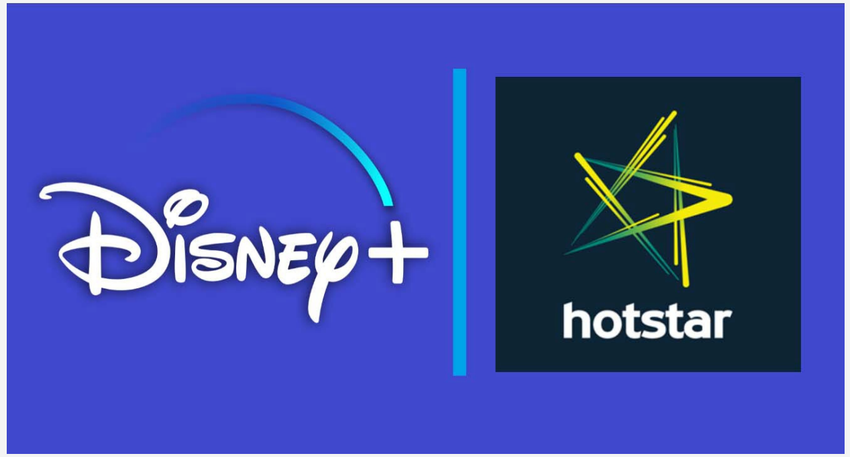


- Introduction
- Problem Statement
- Data Description
- Data Preprocessing
- EDA
- Feature Engineering
- Model Evaluation
- Model Deployment
- Future Developments

Introduction

Hotstar is a platform that has over the 100 million user count, and more than 35,000 hours of content accessible over a variety of genres. As is the case with most major streaming services, using the sheer scale of users and and content to create tailor made recommendations and content for users generates a lot of value for Hotstar.

The goal of this project is to predict user sentiment so that Hotstar can boost content for users and increase viewing.



Some of ways of identifying user sentiments include conducting a survey at regular intervals asking customers their likes, dislikes and preferences, capturing regular feedback regarding user experience etc. But, surveying customers and asking for feedback regularly can prove tedious for the users and hamper the user experience. Hence, we look at an alternate method of deducing user sentiment, by analysing the user behaviour.

Proposed Solution

RATHER THAN TAKING THE DETAILED FEEDBACK FROM THE USER WE JUST ASK THE USER IF THEY HAVE A POSITIVE OR NEGATIVE SENTIMENT REGARDING THE PLATFORM. WE THEN UTILIZE THIS INFORMATION, AS WELL AS THE USER'S WATCH TIME, TO TRY TO UNCOVER CORRELATIONS BETWEEN THESE VARIABLES USING MACHINE LEARNING MODELS.

Data Description

The data for this project consists of below two files

- **train_df.json** - consist of watch time and sentiment data for 200000 users
- **test_df.json** - consist of only watch time for 100000 users

Column Description

Variable	Description
ID	unique identifier variable
titles	titles of the shows watched by the user and watch_time on different titles in the format “title:watch_time” separated by comma, e.g. “JOLLY LLB:23, Ishqbaaz:40”. watch_time is in seconds
genres	same format as titles
cities	same format as titles
tod	total watch time of the user spreaded across different time of days (24 hours format) in the format “time_of_day:watch_time” separated by comma, e.g. “1:454, “17”:5444”
dow	total watch time of the user spreaded across different days of week (7 days format) in the format “day_of_week:watch_time” separated by comma, e.g. “1:454, “6”:5444”
segment	target variable. consider them as interest segments. For modeling, encode pos = 1, neg = 0

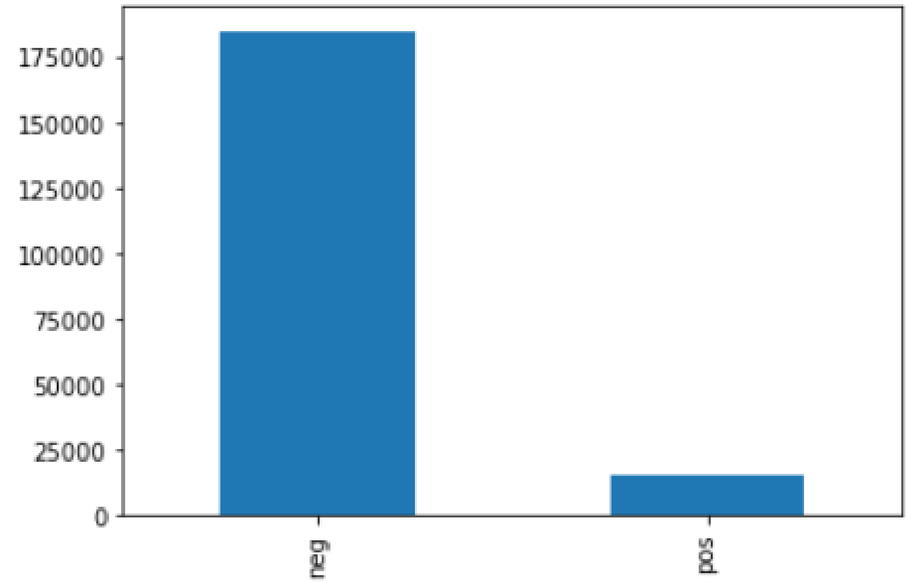
No null values

```
# Checking for the null values in the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 200000 entries, train-121672 to train-126328
Data columns (total 6 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   genres    200000 non-null  object
 1   titles    200000 non-null  object
 2   cities    200000 non-null  object
 3   segment   200000 non-null  object
 4   dow       200000 non-null  object
 5   tod       200000 non-null  object
dtypes: object(6)
memory usage: 14.7+ MB
```

Heavy class imbalance

```
# Checking for the distribution of the target variables
df['segment'].value_counts().plot.bar();
```



Why did we choose this problem?

- The dataset for this problem is a real-world dataset.
- The dataset has 200,000 observations which is huge.
- The dataset is quite unclean and requires a good amount of preprocessing.
- The imbalance in the target variable is high. (92:8)
- The number of features after preprocessing will be huge.

Data Preprocessing

Data before Preprocessing

	genres	titles	cities	segment	dow	tod
train-121672	Drama:6,Cricket:3469,Wildlife:79	Wild Sex:79,Vintage Dhoni Comes Good with 134:...	delhi:3492,mumbai:64	neg	5:482,4:3008,7:64	18:1372,21:1862,17:320
train-121673	Drama:48949,Family:9927,Crime:1901,Romance:165...	Naamkarann:369,Ishq:59,Pardes Mein Hai Mera Di...	ahmedabad:56666,mumbai:26989	neg	1:3257,3:9162,2:12893,5:9237,4:8503,7:25094,6:...	11:7726,10:3521,13:2067,12:3844,20:4608,21:936...
train-121670	Cricket:5715,Family:2777,Drama:20873,LiveTV:4,...	India vs Australia 1st Test Hindi:83,India vs ...	navi mumbai:29400	neg	1:40,3:2700,2:4659,5:7338,4:6641,7:2218,6:5800	11:302,10:1594,13:331,12:133,20:3802,14:3083,2...
train-121671	Romance:77,Drama:122,Cricket:3883	Rangoon:77,India A vs England XI:3883,Kaabil:122	chennai:77,navi mumbai:4005,mumbai:0	neg	1:77,2:3883,5:122	10:122,12:77,15:1800,14:0,16:2083
train-121676	TalkShow:8529	Koffee With Karan:8529	bangalore:5135,krishnarajapura:529,hosur:2864	pos	1:6150,3:2378	20:529,21:1257,22:4930,23:538,19:1268,18:6

- The value for every feature type is a pair of two values - *Feature Subtype* and *user watch time* for that subtype.
- There are multiple such pairs for every user inside every feature.

```
first_row = df_100.iloc[0, :]['cities']
first_row
```

```
'delhi:3492,mumbai:64'
```

```
# First row
first_row_df = pd.DataFrame({i.split(":")[0]:[int(i.split(":")[1])] for i in first_row.split(",")})
first_row_df
```

	delhi	mumbai
0	3492	64

```
'ahmedabad:56666,mumbai:26989'
```

	ahmedabad	mumbai
0	56666	26989

Combining the two data frames we get

```
combined = pd.concat([first_row_df, second_row_df], ignore_index=True, sort=False)
combined
```

	delhi	mumbai	ahmedabad
0	3492.0	64	NaN
1	NaN	26989	56666.0

Combining all the cities for first 100 users we get

user_id	delhi	mumbai	ahmedabad	navi mumbai	chennai	bangalore	krishnarajapura	hosur	gurgaon	nagari	...	dhaka	karachi	madikeri	kochi	peshawar
121672	3492.0	64.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121673	0.0	26989.0	56666.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121670	0.0	0.0	0.0	29400.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121671	0.0	0.0	0.0	4005.0	77.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121676	0.0	0.0	0.0	0.0	0.0	5135.0	529.0	2864.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

5 rows × 33 columns

Final dataframe for the cities of all the 200000 users looks like the one below

user_id	delhi	mumbai	ahmedabad	navi mumbai	chennai	bangalore	krishnarajapura	hosur	gurgaon	nagari	...	lyon	fontaine	saint-nizier-du-moucherotte	orly	geromina
121672	3492.0	64.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121673	0.0	26989.0	56666.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121670	0.0	0.0	0.0	29400.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121671	0.0	0.0	0.0	4005.0	77.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
121676	0.0	0.0	0.0	0.0	0.0	5135.0	529.0	2864.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
126324	0.0	1548.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
126327	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
126326	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
126329	8945.0	0.0	33811.0	0.0	7999.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
126328	0.0	0.0	0.0	49286.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

200000 rows × 1358 columns

- The total no of columns is 1358. That means our users have used the platform in 1358 cities

Doing the same for the Genres feature -

user_id	Drama	Cricket	Wildlife	Family	Crime	Romance	Action	Comedy	LiveTV	TalkShow	...	Table Tennis	Documentary	Tennis	Volleyball	Athletics	Form
121672	6.0	3469.0	79.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
121673	48949.0	0.0	0.0	9927.0	1901.0	16571.0	2064.0	4225.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
121670	20873.0	5715.0	0.0	2777.0	0.0	0.0	14.0	15.0	4.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
121671	122.0	3883.0	0.0	0.0	0.0	77.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
121676	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8529.0	...	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
126324	0.0	3787.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
126327	0.0	5060.0	0.0	0.0	1872.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
126326	25428.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
126329	50749.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
126328	14565.0	17417.0	0.0	0.0	0.0	0.0	41.0	0.0	1662.0	0.0	...	0.0	0.0	0.0	0.0	0.0	

200000 rows × 35 columns

- There are 35 genres for all the 200000 users

Treating the time of the day column similarly -

user_id	6 pm	9 pm	5 pm	11 am	10 am	1 pm	12 pm	8 pm	4 pm	7 pm	...	2 am	9 am	3 pm	2 pm	8 am	7 am	6 am	5 am	3 am
121672	1372.0	1862.0	320.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
121673	4274.0	9360.0	5734.0	7726.0	3521.0	2067.0	3844.0	4608.0	4438.0	3822.0	...	9807.0	2005.0	967.0	2807.0	0.0	0.0	0.0	0.0	0.0
121670	101.0	3430.0	0.0	302.0	1594.0	331.0	133.0	3802.0	0.0	3683.0	...	0.0	4388.0	1640.0	3083.0	201.0	504.0	1874.0	0.0	0.0
121671	0.0	0.0	0.0	0.0	122.0	0.0	77.0	0.0	2083.0	0.0	...	0.0	0.0	1800.0	0.0	0.0	0.0	0.0	0.0	0.0
121676	6.0	1257.0	0.0	0.0	0.0	0.0	0.0	529.0	0.0	1268.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
126324	0.0	2327.0	209.0	0.0	0.0	1108.0	0.0	0.0	11.0	128.0	...	0.0	0.0	17.0	0.0	0.0	0.0	0.0	0.0	0.0
126327	0.0	2128.0	601.0	0.0	0.0	1708.0	0.0	526.0	0.0	154.0	...	423.0	0.0	279.0	1.0	0.0	0.0	0.0	0.0	184.0
126326	234.0	1269.0	1595.0	0.0	1237.0	968.0	1296.0	1120.0	2225.0	390.0	...	0.0	1264.0	387.0	4654.0	0.0	2021.0	3990.0	1540.0	0.0
126329	5023.0	993.0	3892.0	0.0	1.0	9913.0	226.0	1837.0	5615.0	2987.0	...	0.0	0.0	3463.0	5666.0	0.0	0.0	0.0	0.0	0.0
126328	2263.0	6992.0	5387.0	3552.0	1290.0	4147.0	2604.0	4237.0	2272.0	4623.0	...	0.0	1341.0	2648.0	4602.0	573.0	0.0	0.0	0.0	0.0

200000 rows × 24 columns

Treating the day of the week in the same manner -

user_id	Thursday	Wednesday	Saturday	Sunday	Tuesday	Monday	Friday
121672	482.0	3008.0	64.0	0.0	0.0	0.0	0.0
121673	9237.0	8503.0	25094.0	3257.0	9162.0	12893.0	15493.0
121670	7338.0	6641.0	2218.0	40.0	2700.0	4659.0	5800.0
121671	122.0	0.0	0.0	77.0	0.0	3883.0	0.0
121676	0.0	0.0	0.0	6150.0	2378.0	0.0	0.0
...	...	...	...	...	...	...	...
126324	40.0	220.0	2472.0	1108.0	0.0	0.0	0.0
126327	2317.0	0.0	3688.0	948.0	0.0	0.0	0.0
126326	3875.0	4274.0	3284.0	2562.0	3153.0	5009.0	3278.0
126329	2759.0	5122.0	4836.0	11880.0	7673.0	17472.0	1012.0
126328	4460.0	4157.0	7896.0	11425.0	7989.0	6421.0	6928.0

200000 rows × 7 columns

Treating the titles column

user_id	MS Dhoni	Naamkarann	Pardes Mein Hai Mera Dil	Tanhaiyan	Ye Hai Mohabbatein	Ghulaam	Mere Angne Mein	Suhani Si Ek Ladki	Jana Na Dil Se Door	Yeh Rishta Kya Kehlata Hai	...	Savdhaan India	Saath Nibhaana Saathiya	Koi Laut Ke Aaya Hai	Dil Hai Hindustani	Nach Baliye	KI E
121672	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
121673	0	369	5863	16	749	42558	647	7	1	4313	...	0	0	0	0	0	
121670	0	0	0	0	0	0	1	0	0	4	...	0	0	0	0	0	
121671	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
121676	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 24 columns

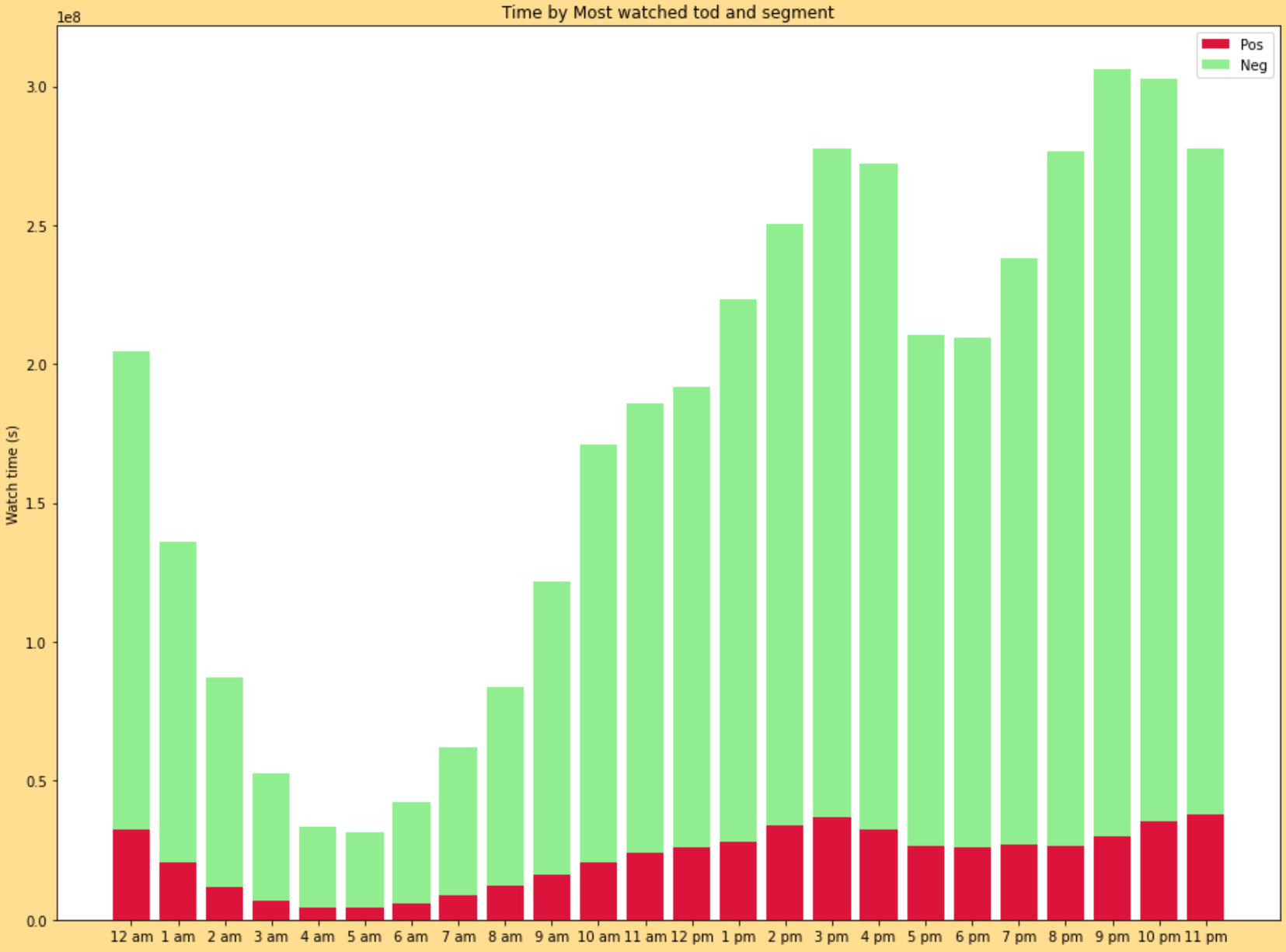
- We have considered only 24 titles that have the maximum watch times that exclude the cricket matches.

Final dataset after combining all features

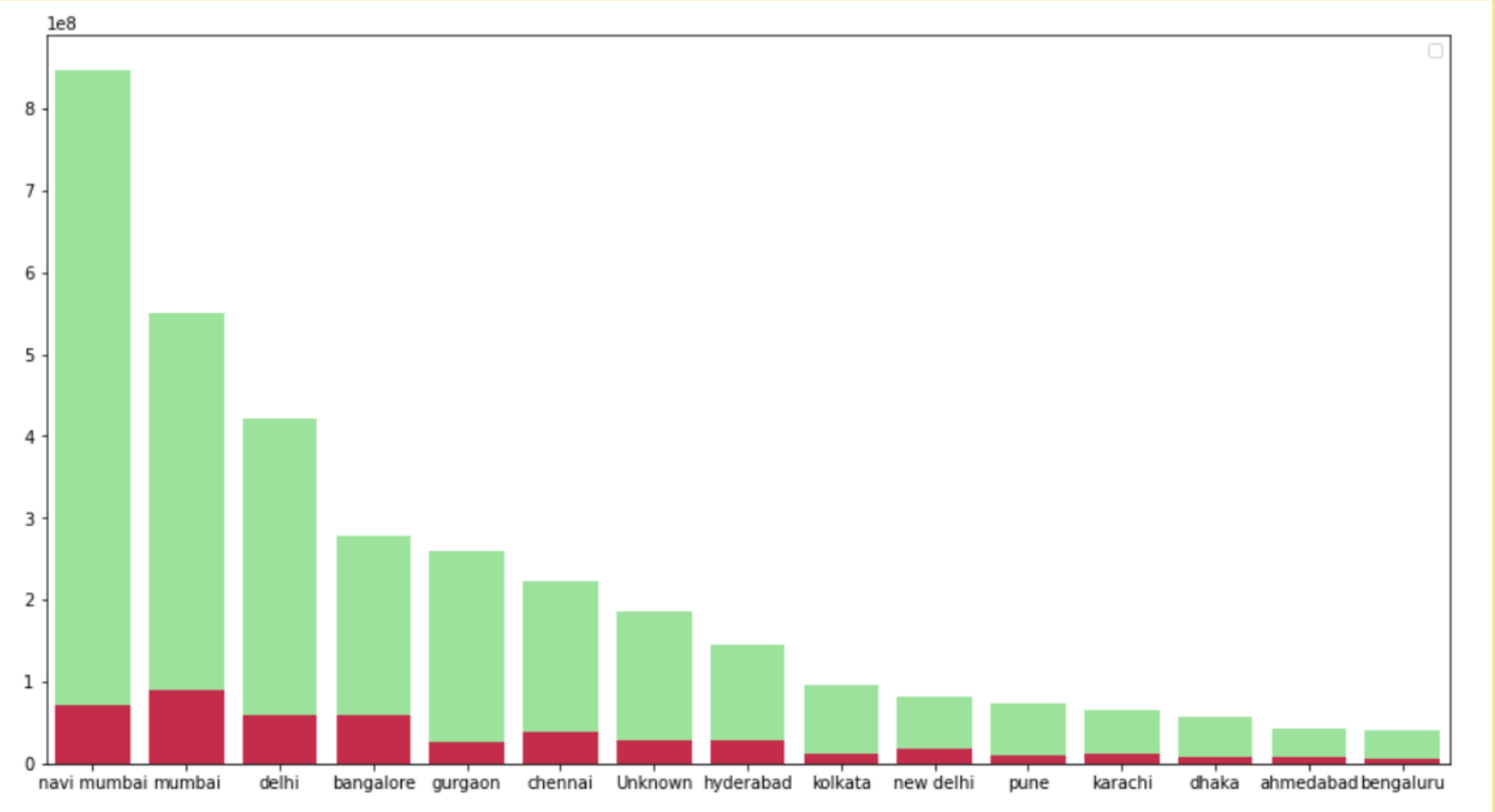
user_id	Drama	Cricket	Wildlife	Family	Crime	Romance	Action	Comedy	LiveTV	TalkShow	...	Savdhaan India	Saath Nibhaana Saathiya	Koi Laut Ke Aaya Hai	Dil Hai Hindustani	Nach Baliye	Khoka Babu	M
121672	6.0	3469.0	79.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	
121673	48949.0	0.0	0.0	9927.0	1901.0	16571.0	2064.0	4225.0	0.0	0.0	...	0	0	0	0	0	0	
121670	20873.0	5715.0	0.0	2777.0	0.0	0.0	14.0	15.0	4.0	0.0	...	0	0	0	0	0	0	
121671	122.0	3883.0	0.0	0.0	0.0	77.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	
121676	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8529.0	...	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
126324	0.0	3787.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	
126327	0.0	5060.0	0.0	0.0	1872.0	0.0	0.0	0.0	0.0	0.0	...	1872	0	0	0	0	0	
126326	25428.0	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	
126329	50749.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0	
126328	14565.0	17417.0	0.0	0.0	0.0	0.0	41.0	0.0	1662.0	0.0	...	0	0	6	14672	0	0	

200000 rows × 1448 columns

EDA

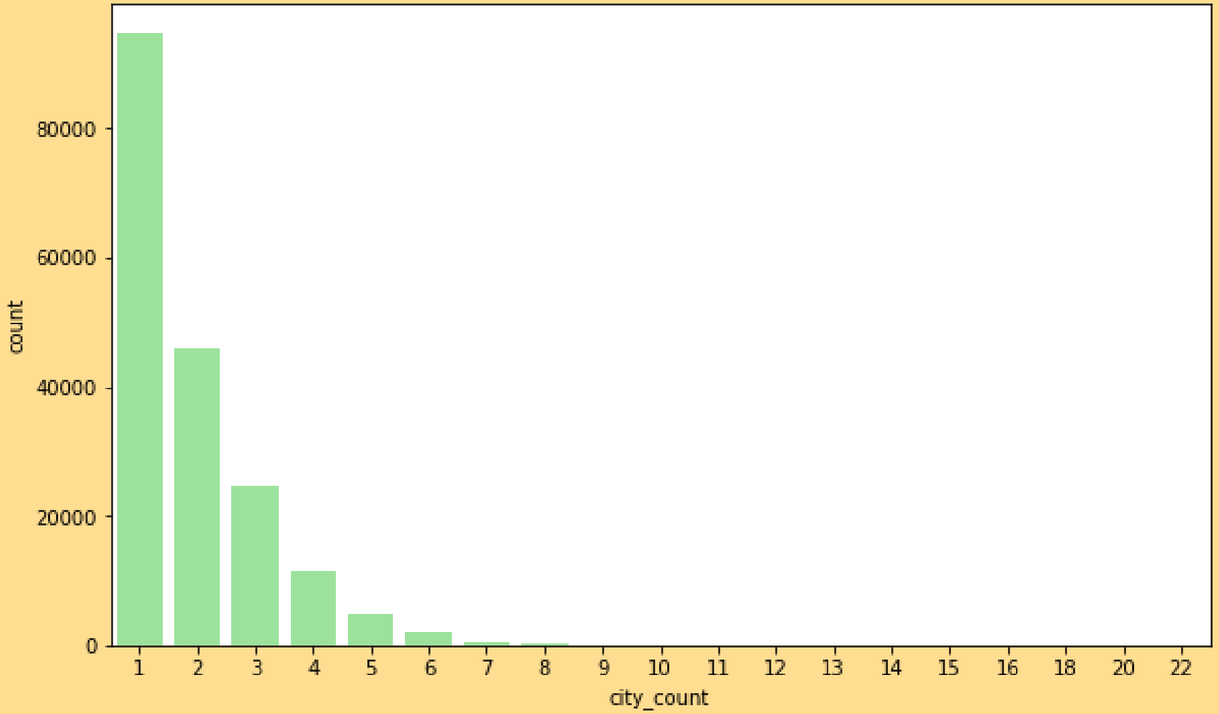

Time by Most watched tod and segment
1e8
Watch time (s)
Pos
Neg
0.0
0.5
1.0
1.5
2.0
2.5
3.0
12 am
1 am
2 am
3 am
4 am
5 am
6 am
7 am
8 am
9 am
10 am
11 am
12 pm
1 pm
2 pm
3 pm
4 pm
5 pm
6 pm
7 pm
8 pm
9 pm
10 pm
11 pm

cities

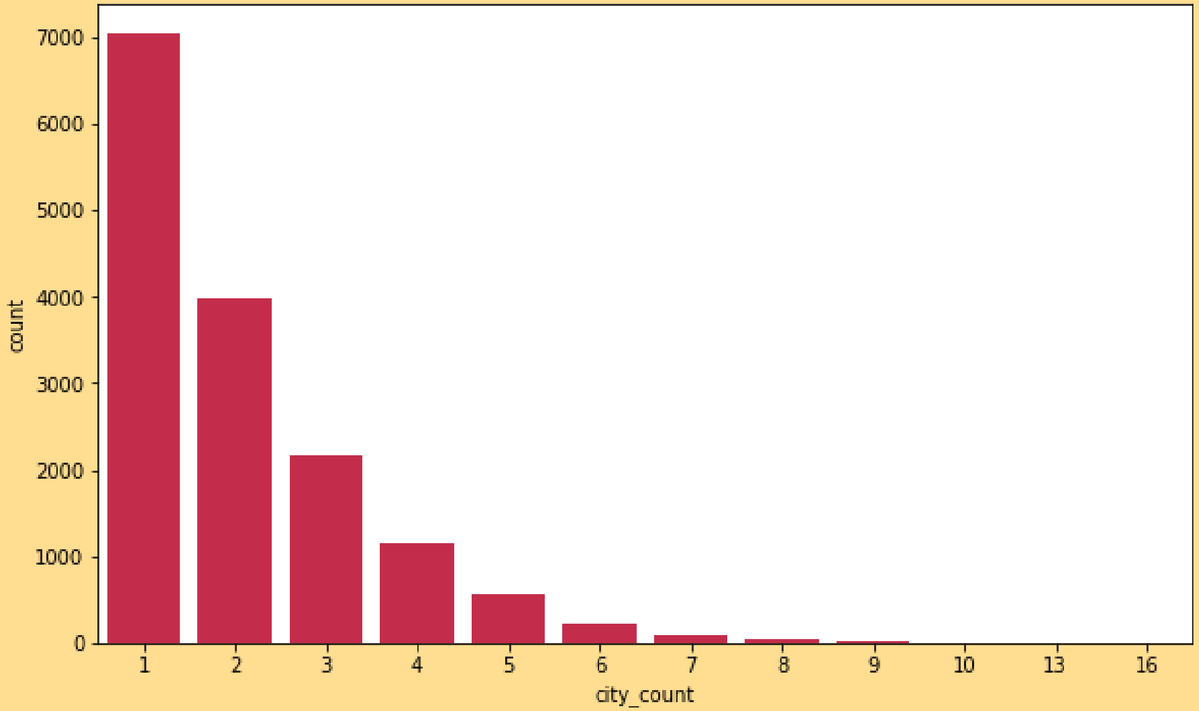

1e8
8
7
6
5
4
3
2
1
0
navi mumbai
mumbai
delhi
bangalore
gurgaon
chennai
Unknown
hyderabad
kolkata
new delhi
pune
karachi
dhaka
ahmedabad
bengaluru

cities

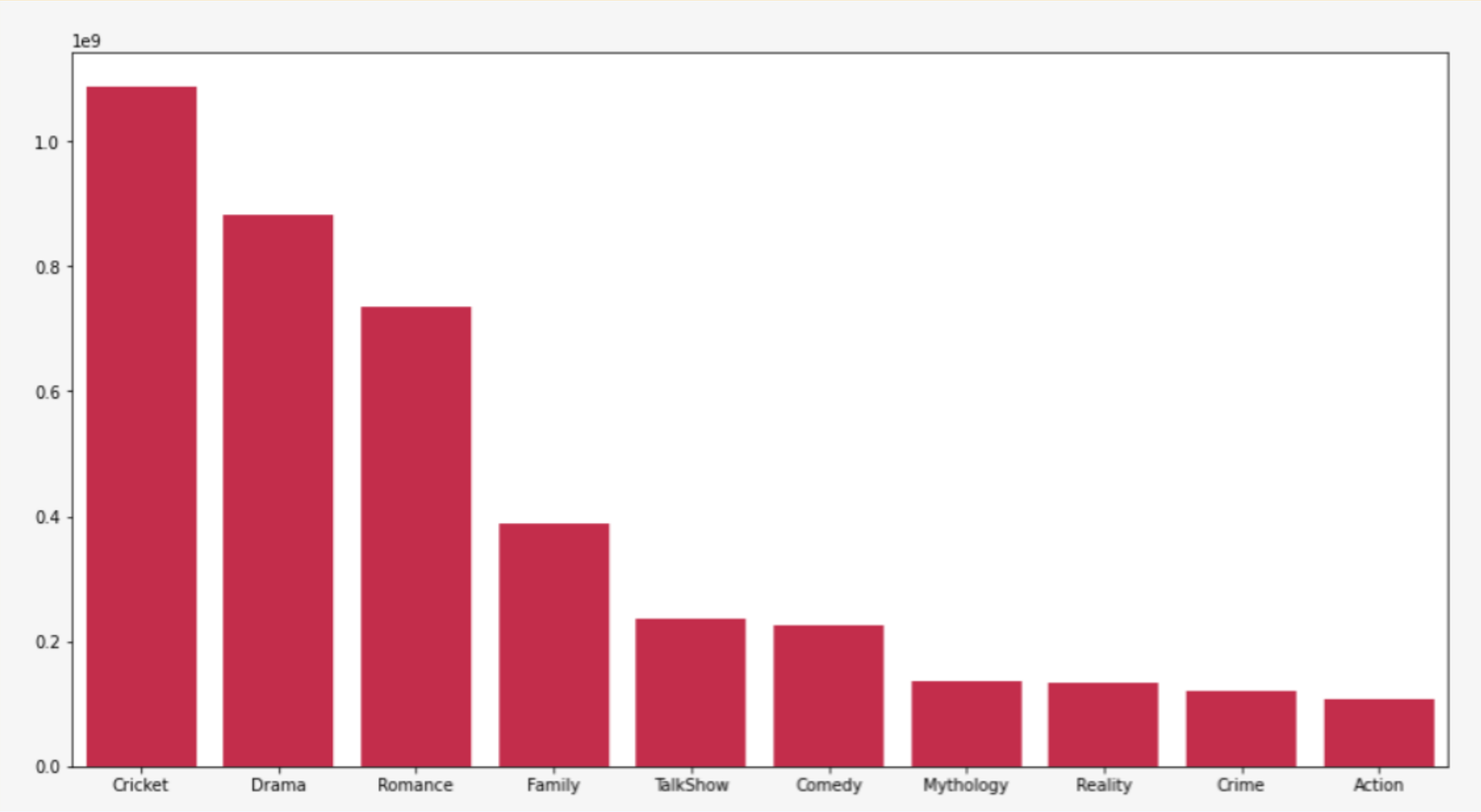
neg



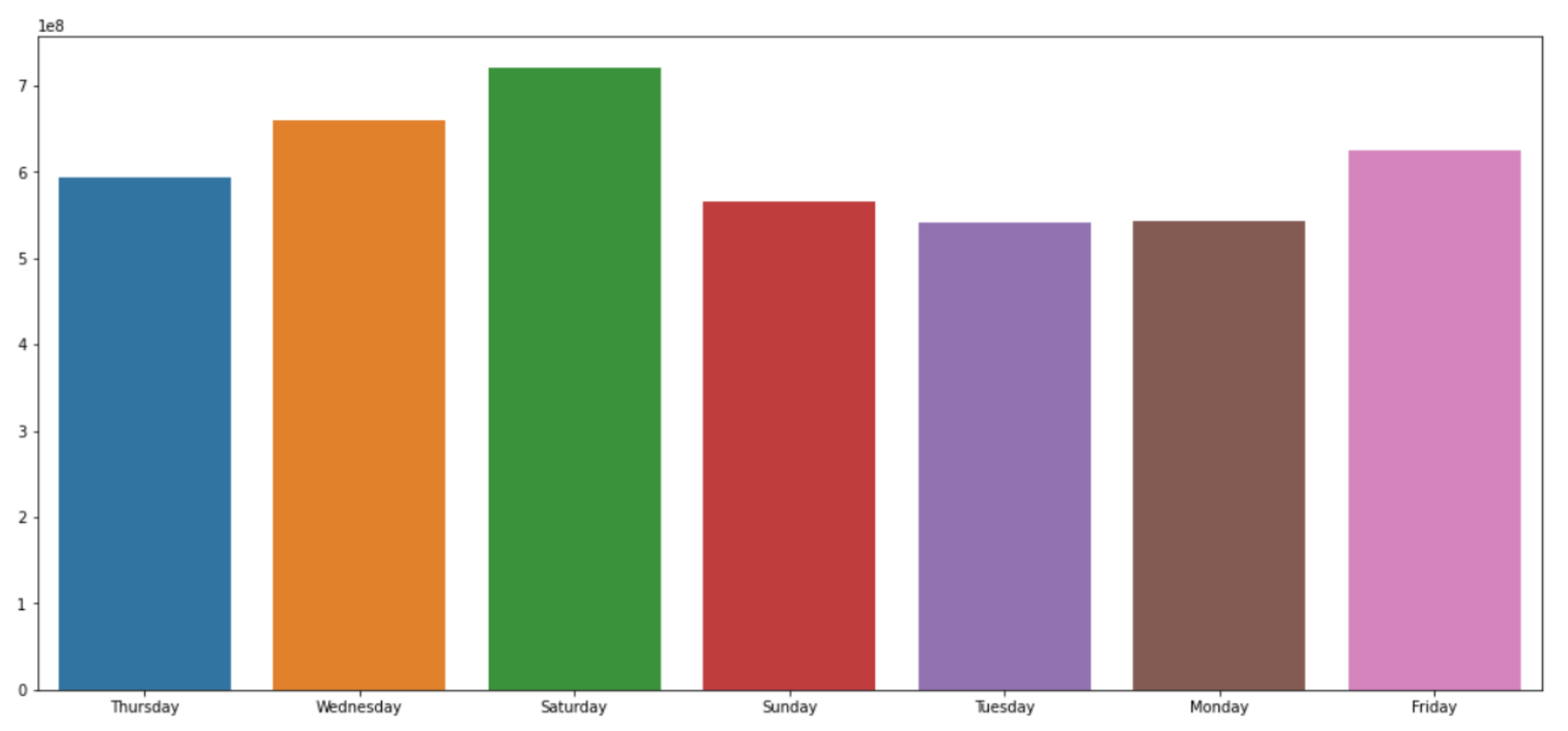
pos



genres


1e9
1.0
0.8
0.6
0.4
0.2
0.0
Cricket
Drama
Romance
Family
TalkShow
Comedy
Mythology
Reality
Crime
Action

dow - day of the week



Feature Engineering

In feature engineering, we have combined a few columns into single column and also added few additional columns for every feature. Let's go through them one by one

- **genres**
  - We have combined all the sports columns like *Badminton, Hockey*, etc. into a single column called *Sport*
  - But the *Cricket* column has been kept as a separate feature since Cricket is one of the most watched genres on the platform.
  - Additionally, we have also added the genre_count for every user as a separate feature.
- **cities**
  - We only consider the top 15 cities that have recorded maximum watch times

- **dow**
  - We have added two columns - one for the *total_watch_time* for all the users and the other for the count of the number of days the user has watched on the platform

- **tod**
  - An additional column representing the number of hours the user has watched on the platform has been created.
  - We only consider the top 5 times of the day that have recorded the maximum user watch times

- **titles**
  - An additional column representing the total number of titles watched by the user has been created.

Our final dataframe after all the preprocessing and feature engineering is represented below. It has 78 features in total excluding the target.

user_id	Drama	Cricket	Wildlife	Family	Crime	Romance	Action	Comedy	LiveTV	TalkShow	...	Savdhaan India	Saath Nibhaana Saathiya	Koi Laut Ke Aaya Hai	Dil Hai Hindustani	Nach Baliye	Khoka Babu
121672	6.0	3469.0	79.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0
121673	48949.0	0.0	0.0	9927.0	1901.0	16571.0	2064.0	4225.0	0.0	0.0	...	0	0	0	0	0	0
121670	20873.0	5715.0	0.0	2777.0	0.0	0.0	14.0	15.0	4.0	0.0	...	0	0	0	0	0	0
121671	122.0	3883.0	0.0	0.0	0.0	77.0	0.0	0.0	0.0	0.0	...	0	0	0	0	0	0
121676	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8529.0	...	0	0	0	0	0	0

5 rows × 78 columns

We are going to apply our Machine Learning models on this dataset

Sampling

To tackle the high imbalance in the data, we have employed oversampling and undersampling techniques.

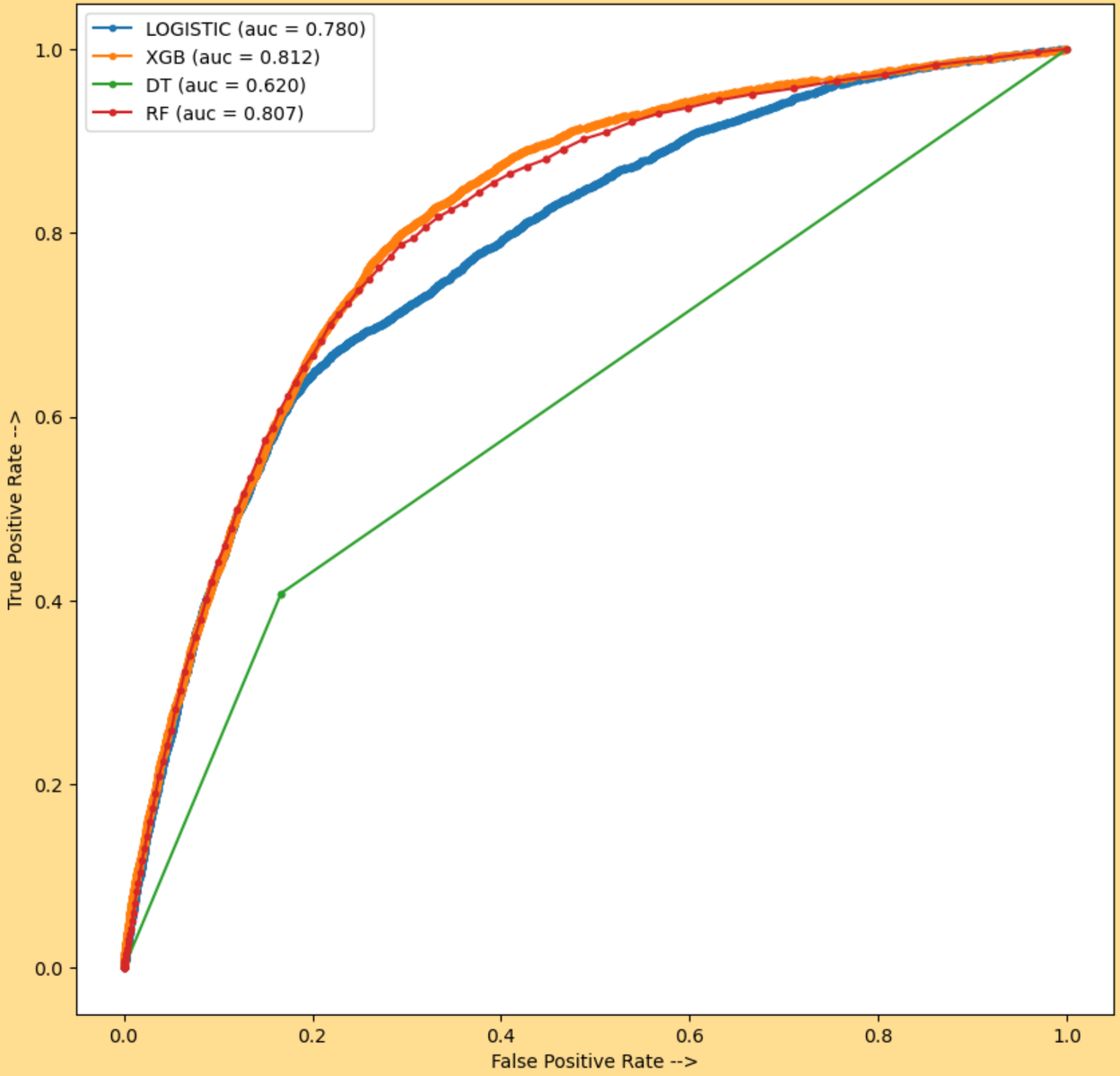
We used SMOTE for generating new examples of minority class and undersampled the majority class.

After the sampling, the ratio of minority class to majority class became 2:5 which was previously 2:23

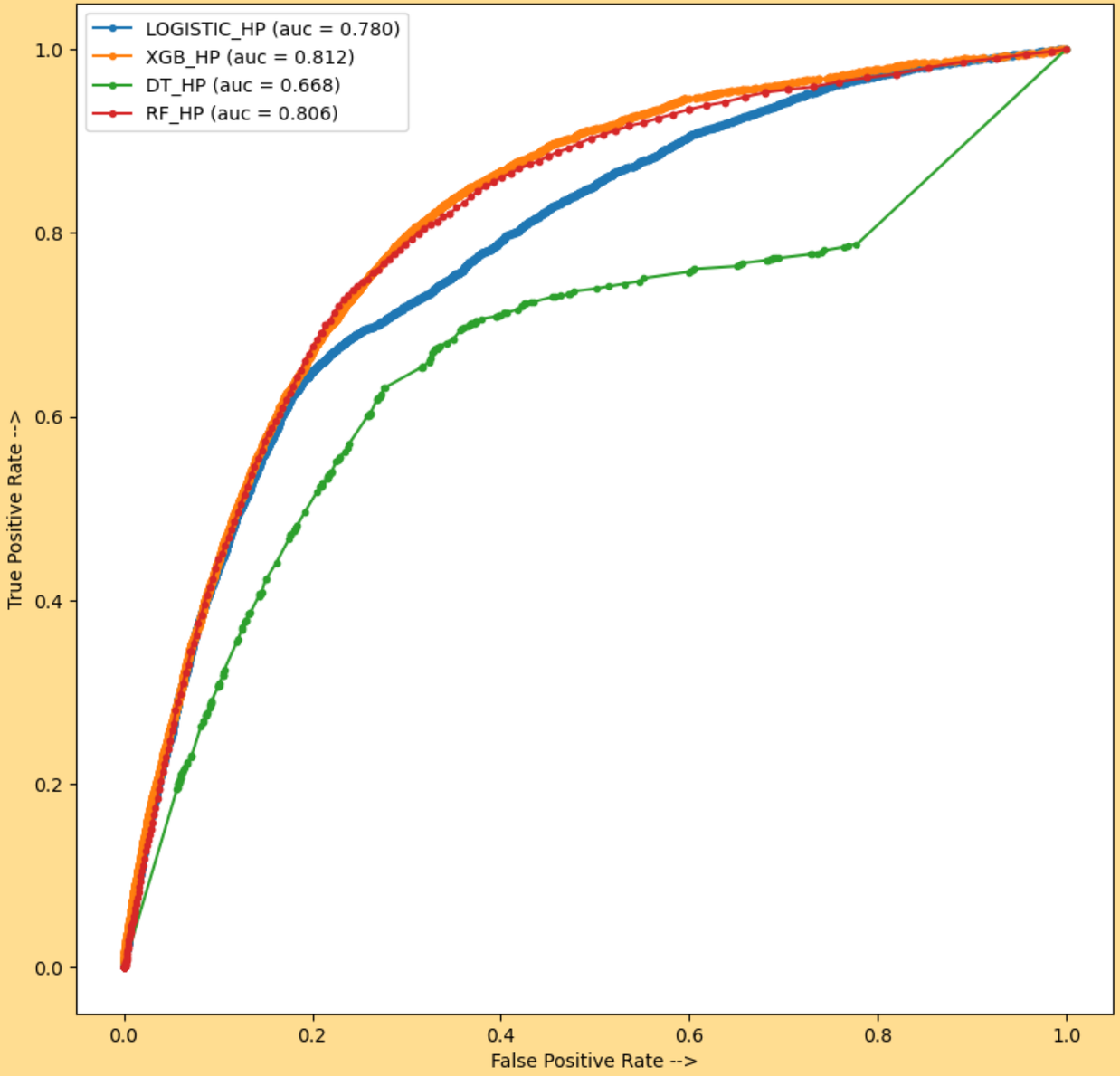
Model Evaluation

	AUC_ROC_Score	
	Without Hyperparameter tuning	With Hyperparameter Tuning
Model		
Logistic Regression	0.7802	0.78029
Decision Tree	0.6204	0.6682
Random Forest	0.8068	0.80566
XGBoost	0.8	**0.8119**

Without hyperparameter tuning



With hyperparameter tuning



Evaluation model on the test data

- Since we have seen in the previous slides that the XGBoost model with hyperparameter tuning gives the highest ROC AUC score and the highest value for the AUC in the plot for TPR vs FPR, we shall choose this model for making predictions on our test data

Deployment Using Streamlit

- The final step in our pipeline is to deploy our model.
- For this purpose, we have used Streamlit.
- ***Streamlit*** is an open-source python framework for building web apps for Machine Learning and Data Science.
- We need to pass the user id of a user from the test data as input and the resulting segment predicted for this user is displayed.
- ***Let's see a live demo on Youtube here***

Future Developments

- Improved feature engineering
- Improvment in the UI to get better insights.

Thank you!