Youtube Streamer Analysis

February 1, 2024

**1 Author :- Harishankar Tiwari**

**1.1 Task :- Youtube Streamer Analysis**

**Intern Career Data scientist Internship**

[1]: **import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**import warnings**

warnings.filterwarnings("ignore")

**from sklearn.preprocessing import** StandardScaler

**from sklearn.cluster import** KMeans

[2]: df = pd.read\_csv("D:\Intern career\youtubers\_df.csv")

[3]: df.head()

[3]: Rank Username Categories Suscribers Country \ 0 1 tseries Música y baile 249500000.0 India 1 2 MrBeast Videojuegos, Humor 183500000.0 Estados Unidos 2 3 CoComelon Educación 165500000.0 Unknown 3 4 SETIndia NaN 162600000.0 India 4 5 KidsDianaShow Animación, Juguetes 113500000.0 Unknown

Visits Likes Comments \

0 86200.0 2700.0 78.0

1 117400000.0 5300000.0 18500.0

2 7000000.0 24700.0 0.0

3 15600.0 166.0 9.0

4 3900000.0 12400.0 0.0

Links

0 http://youtube.com/channel/UCq-Fj5jknLsUf-MWSy…

1 http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u…

2 http://youtube.com/channel/UCbCmjCuTUZos6Inko4…

3 http://youtube.com/channel/UCpEhnqL0y41EpW2TvW…

1

4 http://youtube.com/channel/UCk8GzjMOrta8yxDcKf…

[4]: df.shape

[4]: (1000, 9)

**1.1.1 DATA CLEANING AND EXPLORATION**

[5]: df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Rank 1000 non-null int64

1 Username 1000 non-null object

2 Categories 694 non-null object

3 Suscribers 1000 non-null float64

4 Country 1000 non-null object

5 Visits 1000 non-null float64

6 Likes 1000 non-null float64

7 Comments 1000 non-null float64

8 Links 1000 non-null object

dtypes: float64(4), int64(1), object(4)

memory usage: 70.4+ KB

[6]: df.columns

[6]: Index(['Rank', 'Username', 'Categories', 'Suscribers', 'Country', 'Visits', 'Likes', 'Comments', 'Links'],

dtype='object')

[7]: df.rename(columns={'Suscribers' : 'Subscribers'}, inplace= **True**) [8]: df.columns

[8]: Index(['Rank', 'Username', 'Categories', 'Subscribers', 'Country', 'Visits', 'Likes', 'Comments', 'Links'],

dtype='object')

[9]: df.isnull().sum()

[9]: Rank 0

Username 0

Categories 306

Subscribers 0

Country 0

2

Visits 0

Likes 0

Comments 0

Links 0

dtype: int64

[10]: df['Categories'].fillna('Unknow', inplace= **True**) [11]: df.isnull().sum()

[11]: Rank 0

Username 0

Categories 0

Subscribers 0

Country 0

Visits 0

Likes 0

Comments 0

Links 0

dtype: int64

**1.1.2 CHECK FOR OUTLIERS**

[12]: outliers = ['Subscribers','Visits','Likes','Comments'] fig, axes = plt.subplots(2, 2, figsize=(12, 10)) axes = axes.flatten()

**for** i, col **in** enumerate (outliers):

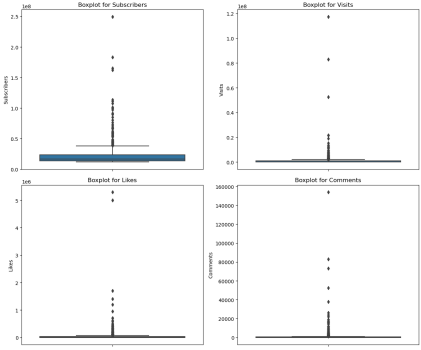
sns.boxplot(y=df[col], ax=axes[i])

axes[i].set\_title(f'Boxplot for **{**col**}**')

plt.tight\_layout()

plt.show()

3



**1.1.3 HANDLING OUTLIERS**

[13]: columns\_to\_check = ['Subscribers', 'Visits', 'Likes', 'Comments'] fig, axes = plt.subplots(2, 2, figsize=(12, 10))

axes = axes.flatten()

**for** i, col **in** enumerate(columns\_to\_check):

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

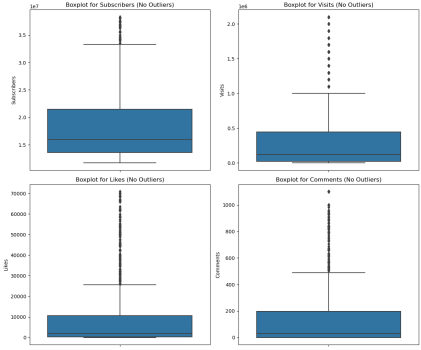
df\_no\_outliers = df[(df[col] >= lower\_bound) & (df[col] <= upper\_bound)] 4

sns.boxplot(y=df\_no\_outliers[col], ax=axes[i])

axes[i].set\_title(f'Boxplot for **{**col**}** (No Outliers)')

plt.tight\_layout()

plt.show()



**1.1.4 Trend Analysis**

[14]: *# Identify trends in categories*

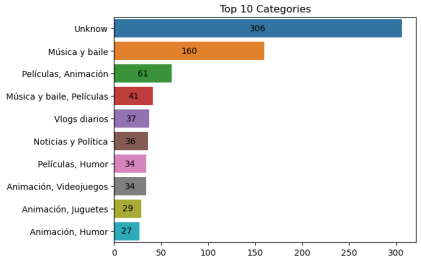
category\_trends = df['Categories'].value\_counts().head(10)

ax= sns.barplot(y= category\_trends.index, x = category\_trends.values) ax.bar\_label(ax.containers[0], label\_type="center")

plt.title('Top 10 Categories')

plt.show()

5



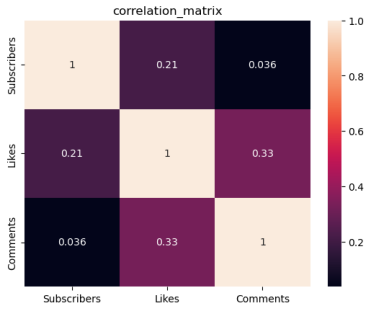
[15]: *#correlation analysis*

correlation\_matrix = df[['Subscribers','Likes','Comments']].corr() sns.heatmap(correlation\_matrix, annot=**True**)

plt.title('correlation\_matrix')

plt.show()

6



**1.1.5 Audience Study**

[16]: *#Analyze distribution of audiences by country*

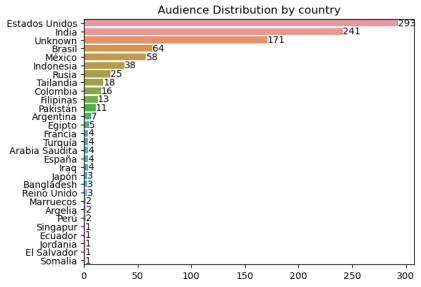
country\_distribution = df['Country'].value\_counts()

yx= sns.barplot(y= country\_distribution.index, x= country\_distribution.values ) yx.bar\_label(yx.containers[0])

plt.title('Audience Distribution by country')

plt.show()

7



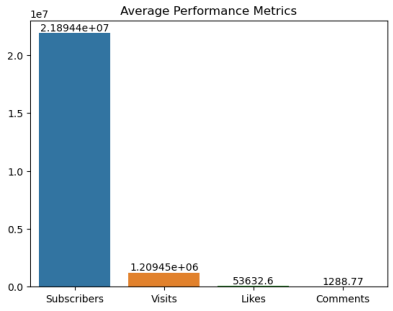
**1.1.6 Performance Metrics**

[17]: average\_metrics = df[['Subscribers', 'Visits', 'Likes', 'Comments']].mean() bx=sns.barplot(x=average\_metrics.index,y=average\_metrics.values) bx.bar\_label(bx.containers[0])

plt.title('Average Performance Metrics')

plt.show()

8



[18]: columns = ['Subscribers', 'Visits', 'Likes', 'Comments'] fig, axes = plt.subplots(2, 2, figsize=(12, 10))

axes = axes.flatten()

**for** i, col **in** enumerate(columns):

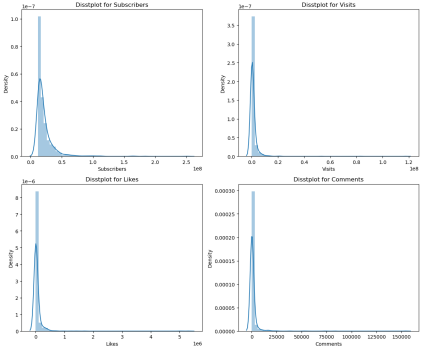
sns.distplot(df[col], ax=axes[i])

axes[i].set\_title(f'Disstplot for **{**col**}**')

plt.tight\_layout()

plt.show()

9



**1.1.7 Content Categories**

[19]: category\_distribution = df['Categories'].value\_counts()

print(category\_distribution)

Unknow 306

Música y baile 160

Películas, Animación 61

Música y baile, Películas 41

Vlogs diarios 37

Noticias y Política 36

Películas, Humor 34

Animación, Videojuegos 34

Animación, Juguetes 29

Animación, Humor 27

Películas 24

Educación 24

Animación 22

10

Videojuegos 19

Videojuegos, Humor 17

Música y baile, Animación 16

Ciencia y tecnología 14

Comida y bebida 12

Humor 10

Juguetes 10

Películas, Juguetes 9

Películas, Videojuegos 8

Deportes 8

Música y baile, Humor 6

Juguetes, Coches y vehículos 4

DIY y Life Hacks 3

Fitness, Salud y autoayuda 3

Videojuegos, Juguetes 3

Animales y mascotas 2

Moda 2

Coches y vehículos 2

Educación, Juguetes 2

Fitness 2

Comida y bebida, Juguetes 1

ASMR, Comida y bebida 1

Animación, Humor, Juguetes 1

Diseño/arte, Belleza 1

Belleza, Moda 1

ASMR 1

Música y baile, Juguetes 1

Diseño/arte, DIY y Life Hacks 1

DIY y Life Hacks, Juguetes 1

Diseño/arte 1

Comida y bebida, Salud y autoayuda 1

Viajes, Espectáculos 1

Juguetes, DIY y Life Hacks 1

Name: Categories, dtype: int64

[20]: performance\_metrics = ['Subscribers', 'Visits', 'Likes', 'Comments'] df.columns = [col.strip() **for** col **in** df.columns]

sns.pairplot(df, vars=performance\_metrics)

plt.suptitle('Pairplot of Performance Metrics')

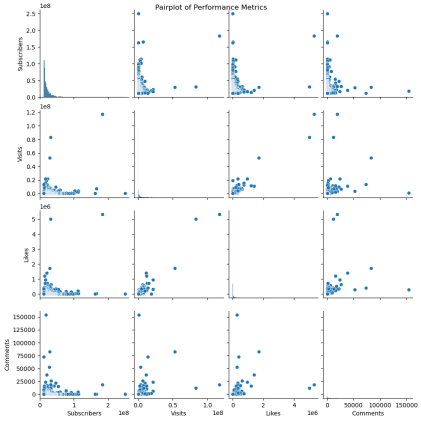
plt.show()

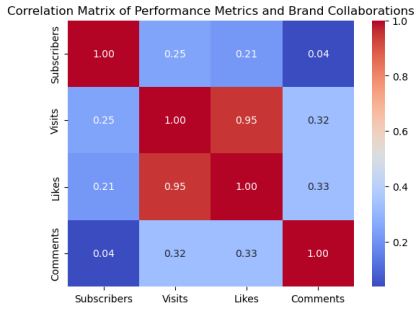
*# Calculate correlation matrix*

correlation\_matrix = df[performance\_metrics + ['Links']].corr() 11

*# Visualize correlation matrix*

sns.heatmap(correlation\_matrix, annot=**True**, cmap='coolwarm', fmt=".2f") plt.title('Correlation Matrix of Performance Metrics and Brand Collaborations') plt.show()

12



[21]: *# Calculate average performance metrics*

average\_metrics = df[['Subscribers', 'Visits', 'Likes', 'Comments']].mean()

*# Identify streamers with above-average performance*

above\_average\_streamers = df[

(df['Subscribers'] > average\_metrics['Subscribers']) &

(df['Visits'] > average\_metrics['Visits']) &

(df['Likes'] > average\_metrics['Likes']) &

(df['Comments'] > average\_metrics['Comments'])

]

*# Display or further analyze above-average streamers*

print(above\_average\_streamers[['Username', 'Subscribers', 'Visits',␣ ↪'Likes','Comments']])

Username Subscribers Visits Likes Comments

1 MrBeast 183500000.0 117400000.0 5300000.0 18500.0 5 PewDiePie 111500000.0 2400000.0 197300.0 4900.0 26 dudeperfect 59700000.0 5300000.0 156500.0 4200.0 34 TaylorSwift 54100000.0 4300000.0 300400.0 15000.0 39 JuegaGerman 48600000.0 2000000.0 117100.0 3000.0

13

43 A4a4a4a4 47300000.0 9700000.0 330400.0 22000.0 58 Mikecrack 43400000.0 2200000.0 183400.0 1800.0 62 KimberlyLoaiza 42100000.0 5300000.0 271300.0 16000.0 64 luisitocomunica 41100000.0 2500000.0 128900.0 1800.0 70 JessNoLimit 39600000.0 1300000.0 73500.0 1600.0 96 TotalGaming093 36300000.0 1500000.0 129400.0 4900.0 98 TechnoGamerzOfficial 35600000.0 6200000.0 341800.0 16500.0 100 markiplier 35500000.0 2100000.0 126500.0 3800.0 122 AboFlah 32700000.0 3300000.0 382000.0 11400.0 123 MRINDIANHACKER 32600000.0 6500000.0 617400.0 26000.0 131 fedevigevani 32000000.0 7700000.0 412200.0 17000.0 132 dream 31900000.0 3300000.0 309200.0 19000.0 136 MrBeast2 31300000.0 83100000.0 5000000.0 11600.0 145 jacksepticeye 30400000.0 1600000.0 83400.0 2300.0 153 DaFuqBoom 29800000.0 52700000.0 1700000.0 82800.0 176 CrazyXYZ 27800000.0 4200000.0 284100.0 8600.0 177 DanTDM 27800000.0 3500000.0 285000.0 52500.0 179 brentrivera 27600000.0 6400000.0 154100.0 5000.0 180 NichLmao 27500000.0 1500000.0 85800.0 1600.0 195 nickiminaj 26100000.0 1600000.0 98300.0 7600.0 206 AlejoIgoa 25700000.0 5700000.0 208400.0 1700.0 207 ZHCYT 25700000.0 2600000.0 127300.0 2200.0 234 rug 24300000.0 3200000.0 85300.0 5100.0 238 alanbecker 24300000.0 7600000.0 582600.0 5900.0 241 juandediospantojaa 24000000.0 3000000.0 133200.0 3600.0 266 DrossRotzank 23100000.0 1700000.0 105900.0 3900.0 272 AmiRodrigueZZ 22900000.0 4300000.0 294400.0 1300.0 278 StokesTwins 22700000.0 11700000.0 235000.0 10000.0 281 SSundee 22700000.0 1700000.0 59800.0 1800.0 282 souravjoshivlogs7028 22700000.0 5600000.0 382300.0 8900.0 288 VillageCookingChannel 22500000.0 21500000.0 321500.0 5900.0 300 alfredolarin 21900000.0 12900000.0 707600.0 2100.0 302 royaltyfam 21900000.0 4700000.0 67000.0 6600.0

**1.1.8 top performing content creator**

[22]: scaler = StandardScaler()

scaled\_metrics= scaler.fit\_transform(df[['Subscribers', 'Visits', 'Likes',␣ ↪'Comments']])

kmeans = KMeans(n\_clusters=2)

df['cluster']= kmeans.fit\_predict(scaled\_metrics)

[23]: top\_performers = df[df['cluster'] == 1]

top\_performers

[23]: Rank Username Categories Subscribers Country \ 1 2 MrBeast Videojuegos, Humor 183500000.0 Estados Unidos

14

136 137 MrBeast2 Vlogs diarios 31300000.0 Estados Unidos

Visits Likes Comments \

1 117400000.0 5300000.0 18500.0

136 83100000.0 5000000.0 11600.0

Links cluster

1 http://youtube.com/channel/UCX6OQ3DkcsbYNE6H8u… 1

136 http://youtube.com/channel/UC4-79UOlP48-QNGgCk… 1

[ ]:

**2 Conclusion**

The initial exploration of the YouTube dataset revealed valuable insights into the structure and characteristics of the top 1000 YouTubers. Key observations include identifying trends in content categories, exploring audience distribution by country, and analyzing performance metrics.

[ ]:

[ ]:

15