YouTube Data Analysis Project

Praveen Manimaran

DSC 207

How To Download Dataset

This dataset has been taken from the Global YouTube Statistics 2023 Dataset on Kaggle (https://www.kaggle.com/datasets/nelgiriyewithana/global-youtube-statistics-2023). You can download this as a csv file and read in the file using pandas.

Why I Chose This Dataset?

For my final project for this class, I decided to select the Global YouTube Statistics 2023 Dataset on Kaggle because it has detailed data regarding the top 995 YouTube subscribers across the world. As a daily YouTube user, I really wanted to get an understanding as to how YouTubers were able to get an extremely high subscriber count and see if there are any patterns in the dataset. By exploring these patterns, I wanted to gain insights into the factors that have led to their success and see if there is potentially a way for others to follow in their footsteps to improve their channel. Although I chose only one dataset for this project, I felt that the dataset was extremely extensive and included several other variables regarding economic factors that could potentially impact a YouTubers' subscriber and view count.

Data Preprocessing

Importing Libraries

In [1]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
from math import sqrt
```

Looking at Our Dataset

In [2]:
```
#read in dataset
youtube_df = pd.read_csv('/Users/praveenmanimaran/Desktop/Global YouTube Statis
youtube_df.head(10) #Look at first 10 rows
```

Out[2]:

	rank	Youtuber	subscribers	video views	category	Title	uploads	Country
0	1	T-Series	245000000	2.280000e+11	Music	T-Series	20082	India
1	2	YouTube Movies	170000000	0.000000e+00	Film & Animation	youtubemovies	1	United States
2	3	MrBeast	166000000	2.836884e+10	Entertainment	MrBeast	741	United States
3	4	Cocomelon	162000000	1.640000e+11	Education	Cocomelon - Nursery Rhymes	966	United States
4	5	SET India	159000000	1.480000e+11	Shows	SET India	116536	India
5	6	Music	119000000	0.000000e+00	NaN	Music	0	NaN
6	7	Kids Diana Show	112000000	9.324704e+10	People & Blogs	ýýý Kids Diana Show	1111	United States
7	8	PewDiePie	111000000	2.905804e+10	Gaming	PewDiePie	4716	Japan
8	9	Like Nastya	106000000	9.047906e+10	People & Blogs	Like Nastya Vlog	493	Russia
9	10	Vlad and Niki	98900000	7.718017e+10	Entertainment	Vlad and Niki	574	United States

10 rows × 28 columns

Variables/Features

There are 995 rows and 28 columns in our dataset. Let us take a look at what each of our features represent!

List of Variables:

rank: Position of the YouTube channel based on the number of subscribers

Youtuber: Name of the YouTube channel

subscribers: Number of subscribers to the channel

video views: Total views across all videos on the channel

category: Category or niche of the channel

Title: Title of the YouTube channel

uploads: Total number of videos uploaded on the channel

Country: Country where the YouTube channel originates

Abbreviation: Abbreviation of the country

channel_type: Type of the YouTube channel (e.g., individual, brand)

video_views_rank: Ranking of the channel based on total video views

country_rank: Ranking of the channel based on the number of subscribers within its country

channel_type_rank: Ranking of the channel based on its type (individual or brand)

video_views_for_the_last_30_days: Total video views in the last 30 days

lowest_monthly_earnings: Lowest estimated monthly earnings from the channel

highest_monthly_earnings: Highest estimated monthly earnings from the channel

lowest_yearly_earnings: Lowest estimated yearly earnings from the channel

highest_yearly_earnings: Highest estimated yearly earnings from the channel

subscribers_for_last_30_days: Number of new subscribers gained in the last 30 days

created_year: Year when the YouTube channel was created

created_month: Month when the YouTube channel was created

created_date: Exact date of the YouTube channel's creation

Gross tertiary education enrollment (%): Percentage of the population enrolled in tertiary education in the country

Population: Total population of the country

Unemployment rate: Unemployment rate in the country

Urban_population: Percentage of the population living in urban areas

Latitude: Latitude coordinate of the country's location

Longitude: Longitude coordinate of the country's location

Drop Columns We Will Not Use

I decided to drop some of the columns in our dataset as I felt that it was not relevant to my data analysis and did not think that it was worth exploring in this project.

In [3]:
```
#Drop columns we wont be needing

columns_to_remove = ['country_rank', 'channel_type_rank', 'lowest_yearly_earni
youtube_df = youtube_df.drop(columns = columns_to_remove )


youtube_df
```

Out[3]:

	rank	Youtuber	subscribers	video views	category	Title	uploads	Co
0	1	T-Series	245000000	2.280000e+11	Music	T-Series	20082	
1	2	YouTube Movies	170000000	0.000000e+00	Film & Animation	youtubemovies	1	
2	3	MrBeast	166000000	2.836884e+10	Entertainment	MrBeast	741	
3	4	Cocomelon	162000000	1.640000e+11	Education	Cocomelon - Nursery Rhymes	966	
4	5	SET India	159000000	1.480000e+11	Shows	SET India	116536	
...	...	...	...	...	...	...	...	
990	991	Natan por Aï¿	12300000	9.029610e+09	Sports	Natan por Aï¿	1200	
991	992	Free Fire India Official	12300000	1.674410e+09	People & Blogs	Free Fire India Official	1500	
992	993	Panda	12300000	2.214684e+09	NaN	HybridPanda	2452	Kir
993	994	RobTopGames	12300000	3.741235e+08	Gaming	RobTopGames	39	S
994	995	Make Joke Of	12300000	2.129774e+09	Comedy	Make Joke Of	62	

995 rows × 14 columns

Quality Issues With Dataset

Let us first take a look at the video views column.

In [4]:
```
youtube_df[['Youtuber', 'video views']].head(5)
```

Out[4]:

	Youtuber	video views
0	T-Series	2.280000e+11
1	YouTube Movies	0.000000e+00
2	MrBeast	2.836884e+10
3	Cocomelon	1.640000e+11
4	SET India	1.480000e+11

We can see that YouTube Movies has a total count of 0 video views. This is because YouTube Movies is actually a channel created by YouTube and it does not keep track of the number of views. There are also a few other rows included in our dataset which include YouTube categories(YouTube sports, YouTube gaming). We only want to work with data that consists of real YouTubers that have view counts.

In [5]:
```
#Fixing values that have 0 video view count
youtube_df = youtube_df[youtube_df['video views'] != 0]
```

Let us now take a look at our highest yearly earnings column.

In [6]:
```
#Need to remove columns for YouTube specific channels/topics because the statis
#This can influence our data exploration significantly!

youtube_df['highest_yearly_earnings'].head(20)
```

Out[6]:
```
0     1.084000e+08
2     6.470000e+07
3     9.480000e+07
4     8.750000e+07
6     3.510000e+07
7     1.900000e+06
8     2.300000e+06
9     2.790000e+07
10    3.860000e+07
11    3.430000e+07
13    2.390000e+07
14    8.600000e-01
15    7.960000e+07
16    5.000000e-02
17    8.100000e+06
19    8.500000e+06
20    2.870000e+07
21    8.190000e+07
22    2.270000e+07
23    0.000000e+00
Name: highest_yearly_earnings, dtype: float64
```

After looking at our highest yearly earnings column, I noticed that some of the values have the wrong units and need to be multiplied to keep the data in our column consistent.

In [7]:
```
#Fix highest_yearly_earnings_column


#For values < 1

condition1 = ((youtube_df['highest_yearly_earnings'] < 1) & (youtube_df['highes
youtube_df.loc[condition1, 'highest_yearly_earnings'] *= 100000000

youtube_df[(youtube_df['highest_yearly_earnings'] < 1)]


#For values < 100
condition2 = ((youtube_df['highest_yearly_earnings'] >=1) & (youtube_df['highes
youtube_df.loc[condition2, 'highest_yearly_earnings'] *= 1000000

#For values < 1000
condition3 = ((youtube_df['highest_yearly_earnings'] >=100) & (youtube_df['high
youtube_df.loc[condition3, 'highest_yearly_earnings'] *= 100000

youtube_df['highest_yearly_earnings'].head(20)
```

Out[7]:
```
0     108400000.0
2      64700000.0
3      94800000.0
4      87500000.0
6      35100000.0
7       1900000.0
8       2300000.0
9      27900000.0
10     38600000.0
11     34300000.0
13     23900000.0
14     86000000.0
15     79600000.0
16      5000000.0
17      8100000.0
19      8500000.0
20     28700000.0
21     81900000.0
22     22700000.0
23            0.0
Name: highest_yearly_earnings, dtype: float64
```

Since some of the values in this column have a value of 0.0, which led me to believe that there is missing data in our dataset that we must handle.

Handling Missing Data

In [8]:
```
#Check for Missing Data
youtube_df.isnull().sum()
```

Out[8]:
```
rank                       0
Youtuber                   0
subscribers                0
video views                0
category                  39
Title                      0
uploads                    0
Country                  113
Abbreviation             113
channel_type              27
video_views_rank           1
highest_yearly_earnings    0
created_year               0
Population               117
dtype: int64
```

We see that there are missing data in 5 columns that we need to address prior to performing our Exploratory Data Analysis. I decided fill in missing values in the highest_yearly_earnings column with the average of the column. For other numerical variables, I thought it would make sense to fill in the value as 0. I also chose to handle missing categorical variables and replace the value with "Unspecified" .

In [9]:
```
### Handling Missing Data

#Impute Values that have 0 (values are essentially missing) in the highest_yea
#the average in the column
```

```
yearly_earnings_mean = youtube_df['highest_yearly_earnings'].mean()


youtube_df['highest_yearly_earnings'] = youtube_df['highest_yearly_earnings'].
youtube_df['highest_yearly_earnings'].fillna(yearly_earnings_mean, inplace=Tru


columns_to_fill_categorical = ['category', 'Country', 'Abbreviation', 'channel

# Fill missing categorical values with 'Unspecified' for the specified columns
youtube_df[columns_to_fill_categorical] = youtube_df[columns_to_fill_categorica

#Fill missing numerical values with 0 for the specified columns
columns_to_fill_numerical = ['video_views_rank', 'Population']
youtube_df[columns_to_fill_numerical] = youtube_df[columns_to_fill_numerical].
```

```
/var/folders/2d/yk1d30kd0xx2v0h_z2t83ly40000gn/T/ipykernel_36085/2706077548.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  youtube_df['highest_yearly_earnings'] = youtube_df['highest_yearly_earning
s'].replace(0, np.nan)
/var/folders/2d/yk1d30kd0xx2v0h_z2t83ly40000gn/T/ipykernel_36085/2706077548.p
y:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  youtube_df['highest_yearly_earnings'].fillna(yearly_earnings_mean, inplace=T
rue)
/var/folders/2d/yk1d30kd0xx2v0h_z2t83ly40000gn/T/ipykernel_36085/2706077548.p
y:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  youtube_df[columns_to_fill_categorical] = youtube_df[columns_to_fill_categor
ical].fillna('Unspecified', inplace=False)
/var/folders/2d/yk1d30kd0xx2v0h_z2t83ly40000gn/T/ipykernel_36085/2706077548.p
y:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  youtube_df[columns_to_fill_numerical] = youtube_df[columns_to_fill_numerica
l].fillna(0, inplace=False)
```

In [10]:
```
youtube_df.isnull().sum()
```

Out[10]:
```
rank                      0
Youtuber                  0
subscribers               0
video views               0
category                  0
Title                     0
uploads                   0
Country                   0
Abbreviation              0
channel_type              0
video_views_rank          0
highest_yearly_earnings   0
created_year              0
Population                0
dtype: int64
```

We see that there are no more missing values.

Rename Column Names

Let us now rename our columns so that it will be easier to work with!

In [11]:
```
#Rename Columns to Neat Names
new_column_names = {
    'rank': 'Rank',
    'subscribers': 'Subscribers',
    'video views': 'VideoViews',
    'category': 'Category',
    'uploads': 'Uploads',
    'channel_type': 'ChannelType',
    'video_views_rank': 'VideoViewsRank',
    'highest_yearly_earnings': 'HighestYearlyEarnings',
    'created_year': 'YearCreated'
}

youtube_df.rename(columns=new_column_names, inplace=True)
youtube_df.head()
```

```
/var/folders/2d/yk1d30kd0xx2v0h_z2t83ly40000gn/T/ipykernel_36085/2493049606.p
y:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  youtube_df.rename(columns=new_column_names, inplace=True)
```

Out[11]:

	Rank	Youtuber	Subscribers	VideoViews	Category	Title	Uploads	Country	A
0	1	T-Series	245000000	2.280000e+11	Music	T-Series	20082	India	
2	3	MrBeast	166000000	2.836884e+10	Entertainment	MrBeast	741	United States	
3	4	Cocomelon	162000000	1.640000e+11	Education	Cocomelon - Nursery Rhymes	966	United States	
4	5	SET India	159000000	1.480000e+11	Shows	SET India	116536	India	
6	7	Kids Diana Show	112000000	9.324704e+10	People & Blogs	ýýý Kids Diana Show	1111	United States	

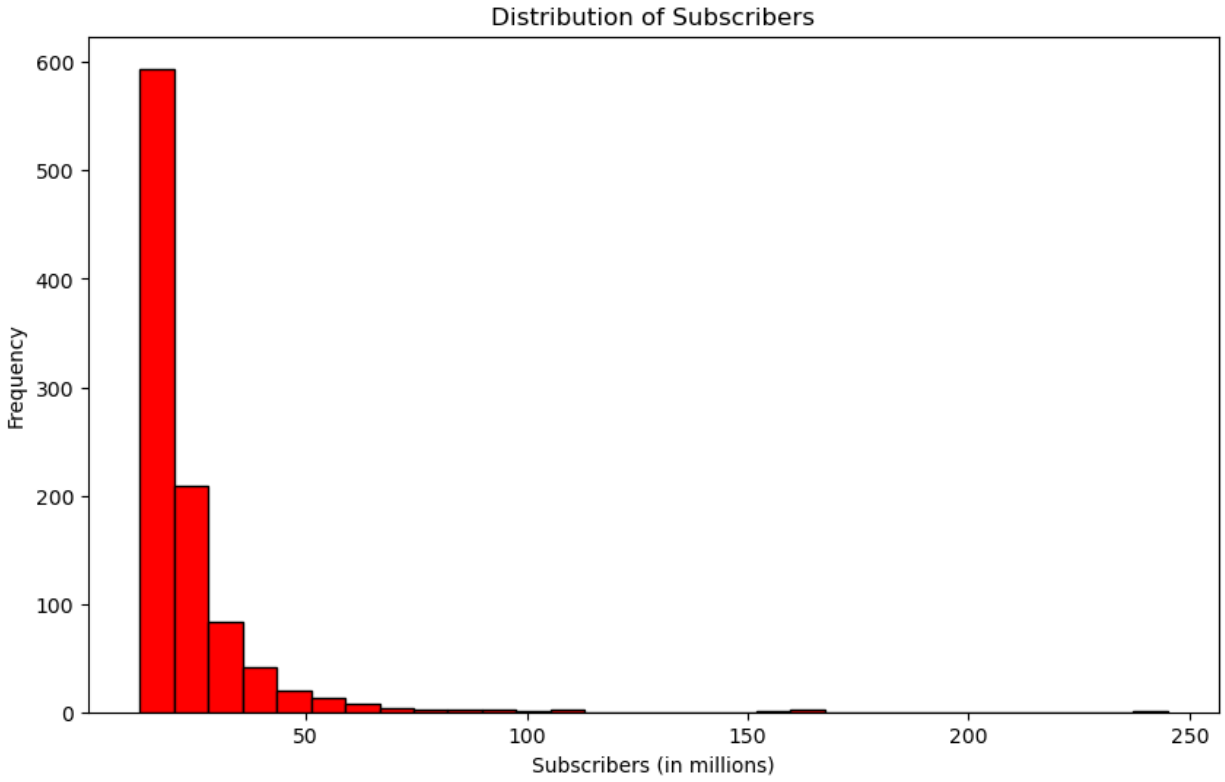
We are now ready to start our Exploratory Data Analysis!

Exploratory Data Analysis

Let us now see how our data is distributed.

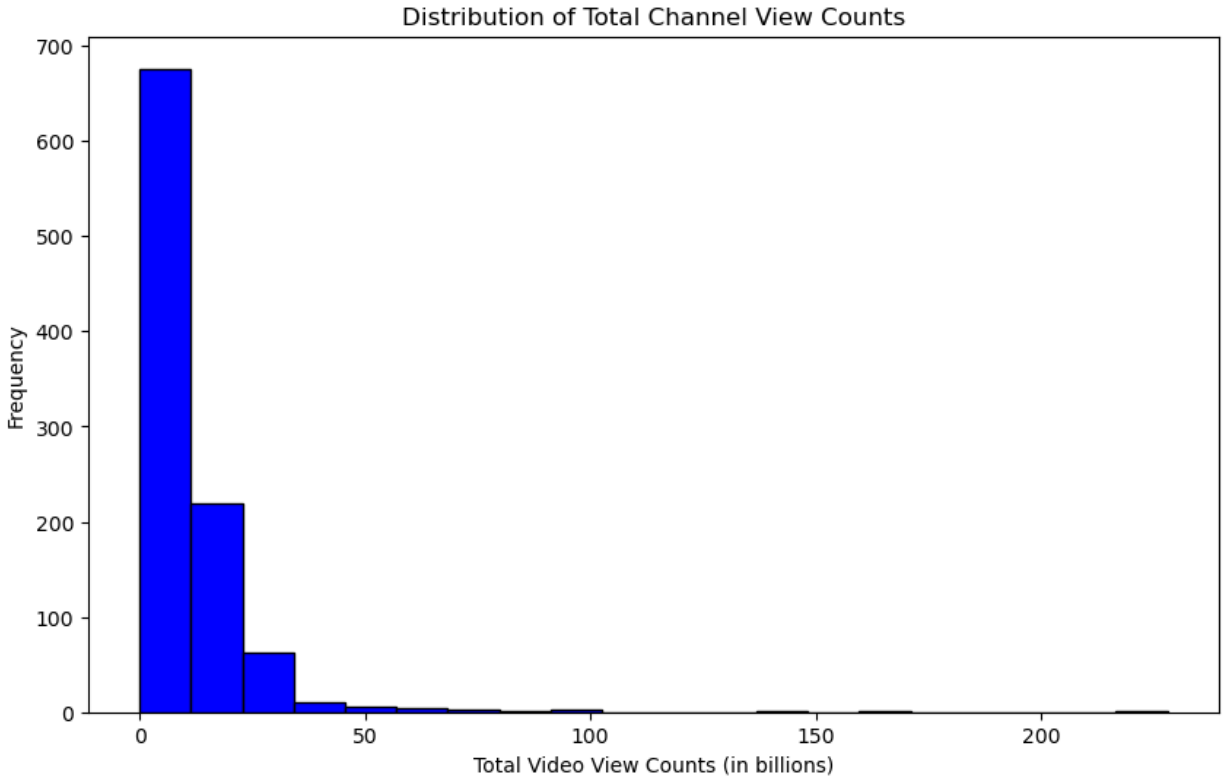
In [12]:
```
#Plot distribution of subscribers and uploads and highest yearly earnings


# Plotting the distribution of subscribers using a histogram
# Convert subscribers to millions to make it easier to read
youtube_df.loc[:, 'Subscribers_Millions'] = youtube_df['Subscribers'] / 1e6

plt.figure(figsize=(10, 6))
plt.hist(youtube_df['Subscribers_Millions'], bins=30, color='red', edgecolor='
plt.title('Distribution of Subscribers')
plt.xlabel('Subscribers (in millions)')
plt.ylabel('Frequency')
plt.show()
```

```
/var/folders/2d/yk1d30kd0xx2v0h_z2t83ly40000gn/T/ipykernel_36085/1894208527.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  youtube_df.loc[:, 'Subscribers_Millions'] = youtube_df['Subscribers'] / 1e6
```



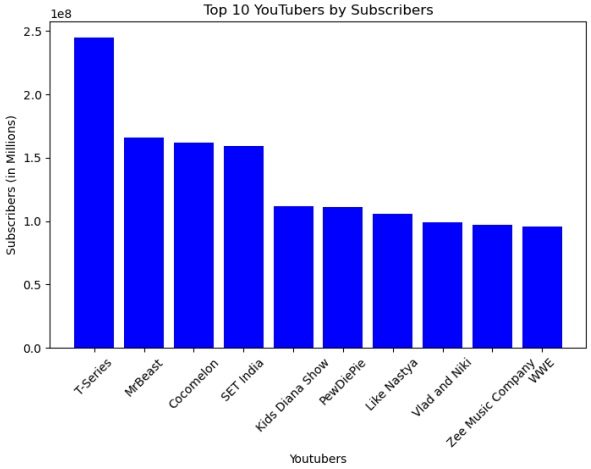
We can see that a majority of the data lies in the range of 10 to 50 million with very few channels outside of this range(Skewed to the right). This makes sense because there are not that many channels on YouTube that have over 50 million subscribers.

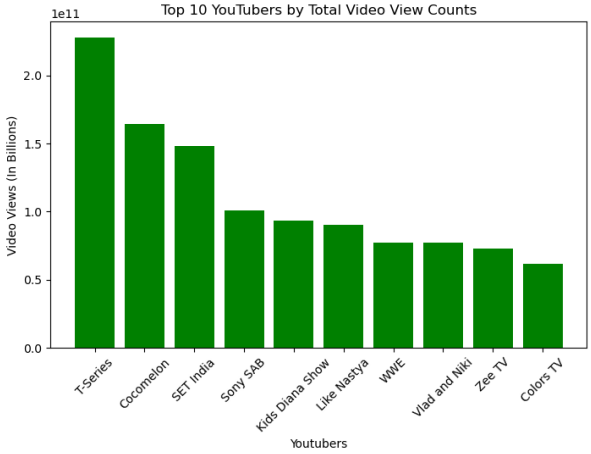
In [13]:
```
# Plotting the distribution of video views using a histogram

plt.figure(figsize=(10, 6))
plt.hist(youtube_df['VideoViews'] /1e9, bins=20, color='blue', edgecolor='blacl
plt.title('Distribution of Total Channel View Counts')
plt.xlabel('Total Video View Counts (in billions)')
plt.ylabel('Frequency')
plt.show()
```



The distribution of total video view counts is also skewed to the right showing us that most of the channels are not able to generate more then 50 billion total views.

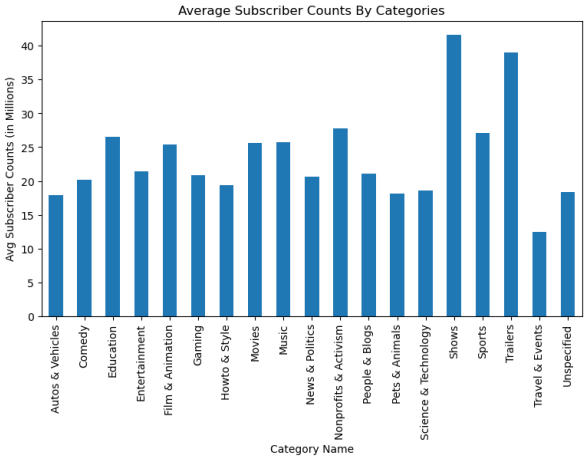
In [14]:
```
# Plot Channels with Top 10 Subscribers and Top 10 Video Counts Side by Side

fig, axes = plt.subplots(1, 2, figsize=(18, 5))

# Select the top 10 Subscribers
top_ten_subs = youtube_df.sort_values('Subscribers', ascending=False).head(10)

# Plot for Top 10 Subscribers
axes[0].bar(top_ten_subs['Youtuber'], top_ten_subs['Subscribers'], color = 'blu
axes[0].set_ylabel('Subscribers (in Millions)')
axes[0].set_xlabel('Youtubers')
axes[0].set_title('Top 10 YouTubers by Subscribers')
axes[0].tick_params(axis='x', rotation=45)

# Select the top 10 Video View Counts
top_ten_views = youtube_df.sort_values('VideoViews', ascending=False).head(10)

# Plot for Top 10 Video Views
axes[1].bar(top_ten_views['Youtuber'], top_ten_views['VideoViews'], color = 'g
axes[1].set_ylabel('Video Views (In Billions)')
axes[1].set_xlabel('Youtubers')
axes[1].set_title('Top 10 YouTubers by Total Video View Counts')
axes[1].tick_params(axis='x', rotation=45)

plt.show()
```

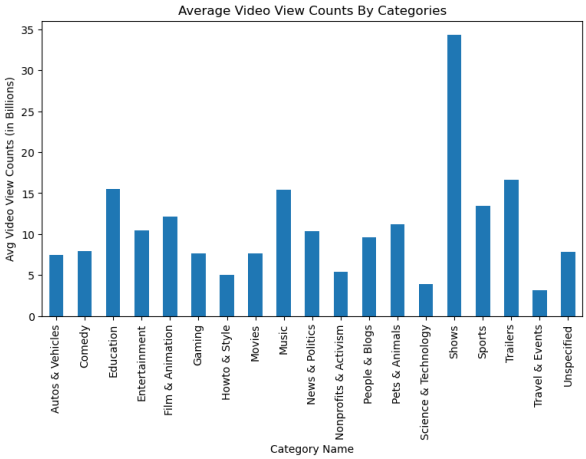




From the side by side bar charts, we can see that although MrBeast and PewDiePie are among the channels that are in the top 10 subscribed to on YouTube, they were not able to make the top 10 in total video view counts. We can also notice that there were channels outside the top 10 subscribed(Zee TV and Colors TV) that have a high engagement with their viewer. Being raised in India, I am able to recognize that most of these channels are from India which indicates that we should maybe try to explore later if there is a relationship between country and video view counts.

In [15]:
```
# Plot Categories by Subscribers and Video Counts Side by Side

fig, axes = plt.subplots(1, 2, figsize=(20, 5))

category_subs = (youtube_df.groupby('Category')['Subscribers'].mean())/1e6

category_subs.plot(kind = 'bar', ax = axes[0])
axes[0].set_ylabel('Avg Subscriber Counts (in Millions)')
axes[0].set_xlabel('Category Name')
axes[0].set_title('Average Subscriber Counts By Categories')

category_video_views = (youtube_df.groupby('Category')['VideoViews'].mean())/1

category_video_views.plot(kind = 'bar', ax = axes[1])
axes[1].set_ylabel('Avg Video View Counts (in Billions)')
axes[1].set_xlabel('Category Name')
axes[1].set_title('Average Video View Counts By Categories')

plt.show()
```



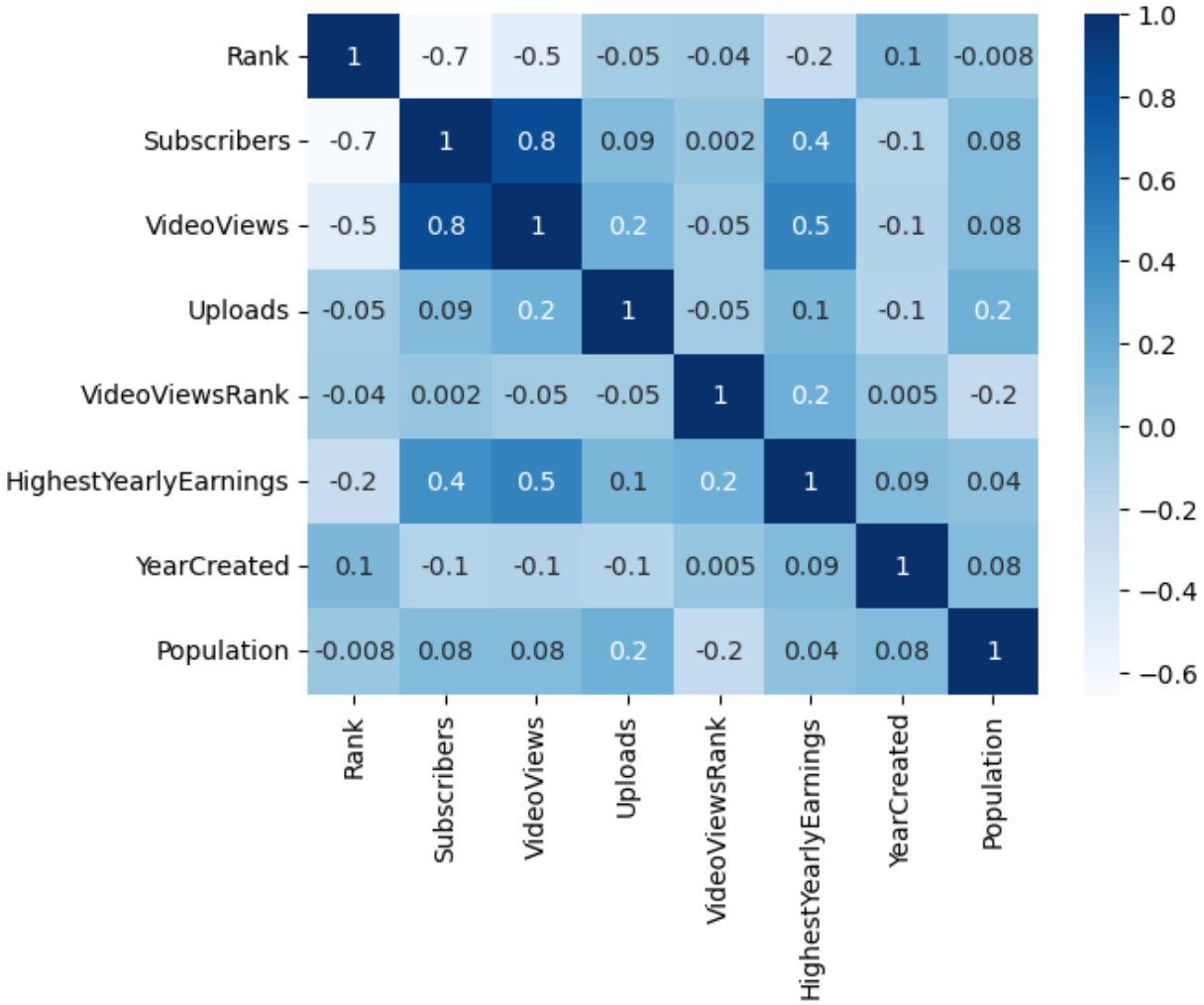


The categories that have the highest average subscriber counts are Shows, Trailers, Nonprofits and Activism, Education, and Sports. The categories that have the highest average video view counts is Sports, Trailers, Education, and Music. We can easily see that the category Shows has a relatively large number of subscribers and total video view counts compared to the other categories. This might be related to the fact that a lot of the top subscribed YouTube channels are from India and TV shows(serials) are extremely popular in India.

Correlation Matrix

To see if any variables are connected to one another and have influence on each other, we can use a correlation matrix.

In [16]:
```
#Correlation Matrix

df_2 = youtube_df.drop(columns = ['Category', 'Youtuber', 'Title', 'Country',
corr = df_2.corr()
#corr
heatmap = sns.heatmap(corr, annot=True, cmap="Blues", fmt='.1g')
```



From the correlation matrix, we can see that the number of subscribers is strongly positively correlated to total video views which makes sense because channels that have the most subscribers are the most popular YouTubers and therefore have more people watching their videos. We can also see a slight positive correlation with Highest Yearly Earnings and the Total Video Views which also makes sense since channels make their money off their views, so higher views means the channel would make more money. I am very suprised to see that the number of uploads does not have much correlation with video view counts. This is worth exploring further!

Research Questions

1. How does the number of uploads by a YouTube channel affect the channel's subscriber count, video view count, and their highest yearly income?

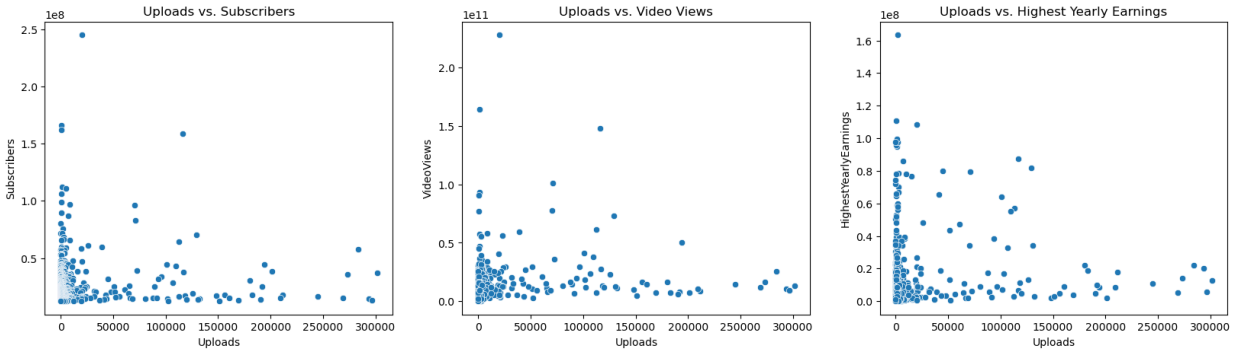
2. Is there a relationship between a channel's country of origin and the total number of views? Also see the influence of a country's population size on view count?

3. Can we build a model that can accurately predict a channel's subscribers count?

#1. How does the number of uploads by a YouTube channel affect the channel's subscriber count, video view count, and their highest yearly income?

I thought that a good way to see the relationship between these variables would be through a scatter plot as it would allow us visually to see if a correlation amongst subscriber count, video view count, and the highest yearly income.

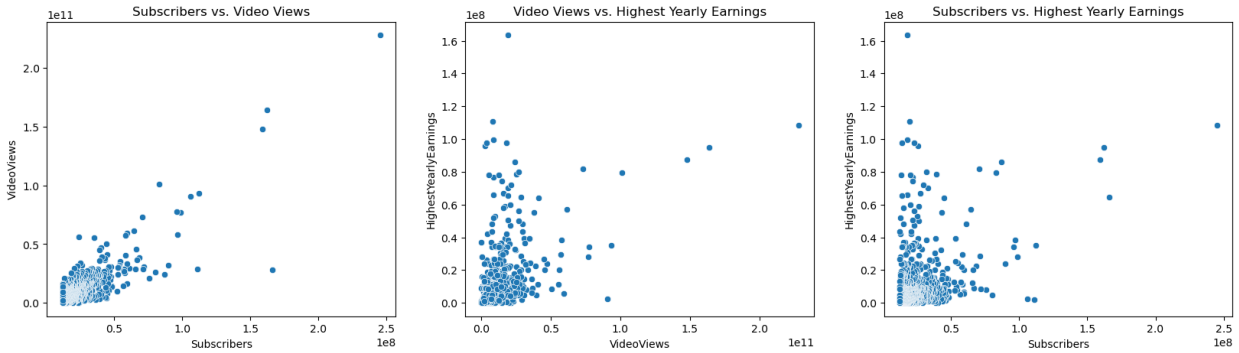
In [17]:
```
#Create 3 side by side plots

fig, axes = plt.subplots(1, 3, figsize=(20, 5))

#Plot scatter plot between uploads and subscriber count
sns.scatterplot(data=youtube_df, x="Uploads", y="Subscribers", ax=axes[0])
axes[0].set_title('Uploads vs. Subscribers')

#Plot scatter plot between uploads and video view count
sns.scatterplot(data=youtube_df, x="Uploads", y="VideoViews", ax=axes[1])
axes[1].set_title('Uploads vs. Video Views')

#Plot scatter plot between uploads and highest yearly income
sns.scatterplot(data=youtube_df, x="Uploads", y="HighestYearlyEarnings", ax=ax
axes[2].set_title('Uploads vs. Highest Yearly Earnings')

plt.show()
```



From the plots above, we can see that the the number of uploads does not seem to have much correlation to the number of subscribers, video views, or highest yearly earnings.

In [18]:
```
#Create 3 side by side plots
fig, axes = plt.subplots(1, 3, figsize=(20, 5))

#Plot scatter plot between uploads and subscriber count
sns.scatterplot(data=youtube_df, x="Subscribers", y="VideoViews", ax=axes[0])
axes[0].set_title('Subscribers vs. Video Views')

#Plot scatter plot between uploads and video view count
sns.scatterplot(data=youtube_df, x="VideoViews", y="HighestYearlyEarnings", ax=
axes[1].set_title('Video Views vs. Highest Yearly Earnings')
```

```
#Plot scatter plot between uploads and highest yearly income
sns.scatterplot(data=youtube_df, x="Subscribers", y="HighestYearlyEarnings", ax
axes[2].set_title('Subscribers vs. Highest Yearly Earnings')

# Show plot
plt.show()
```

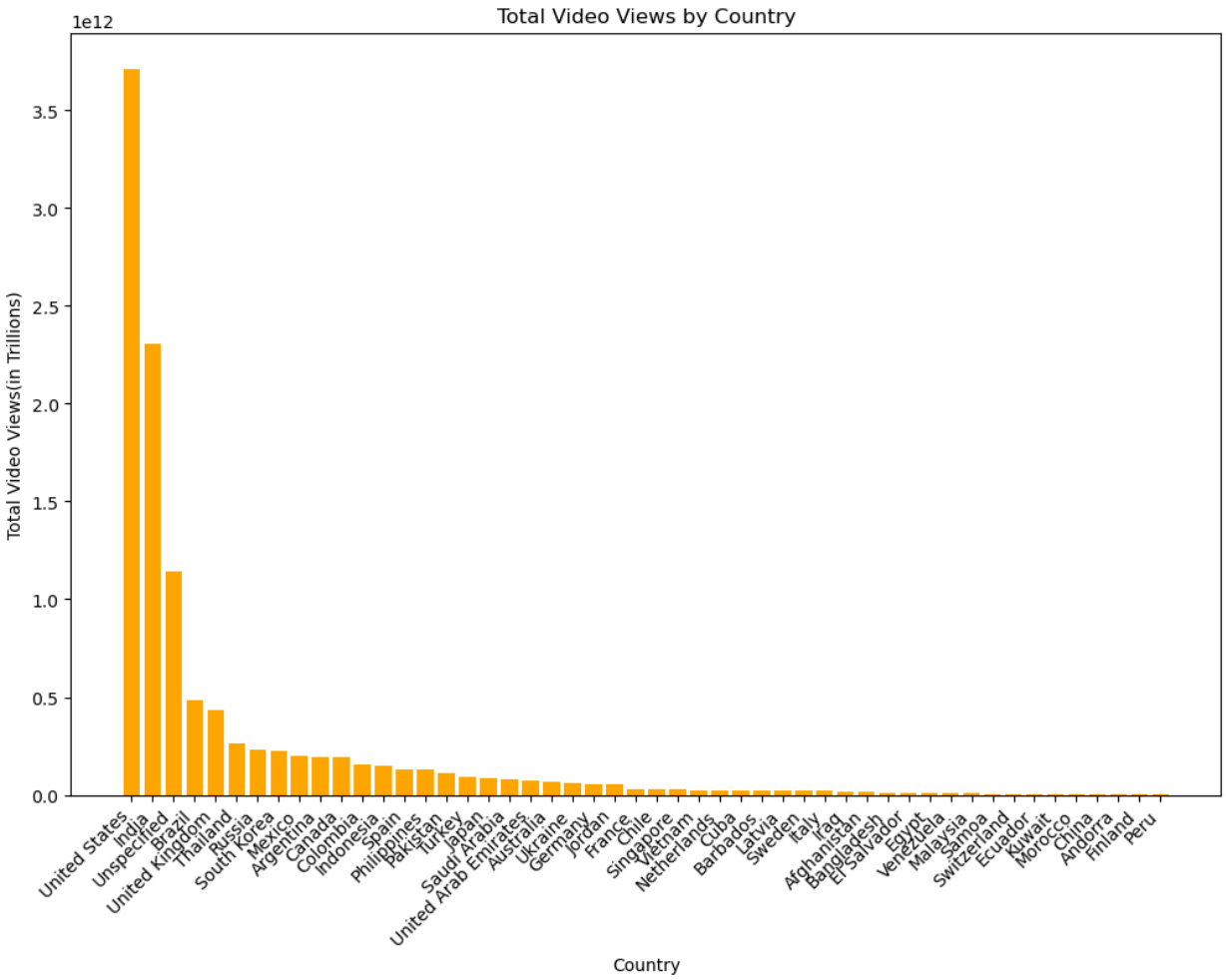


It appears to be the case that subscribers and video views have a strong positive correlation and that video views and highest yearly earnings have a slight positive correlation. Also, subscribers has a slight positive corelation with the highest yearly earnings column. It is import to note that most of the data points are grouped tightly together at the bottom left of the graph because the distributions of subscribers and video views are heavily skewed to the right.

#2. Is there a relationship between a channel's country of origin and the total number of views? If so, do you see an influence of a country's population size on view count?

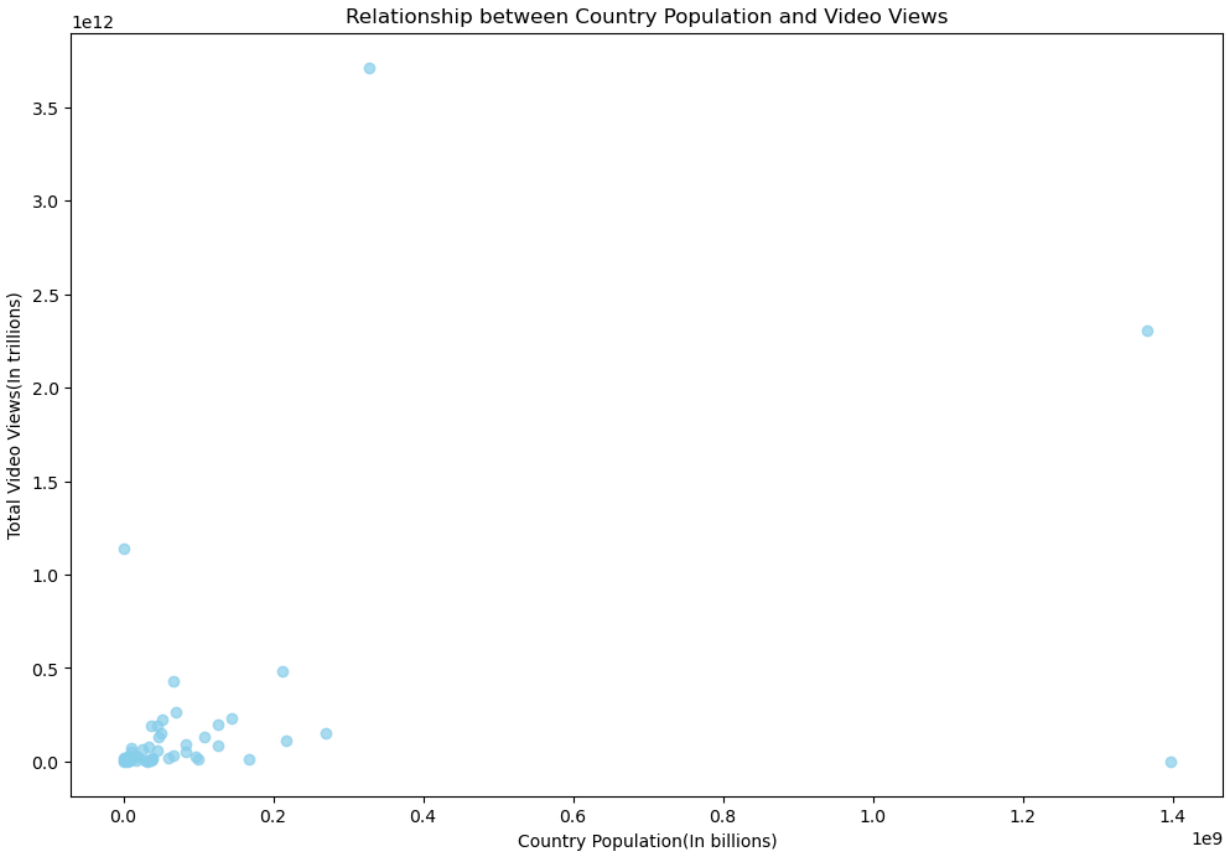
I think that a good way to answer this question would be to group by country and to sum up the total video views for each country and use a bar chart to showcase this visually.

In [19]:
```
# Group by 'Country' and sum up the 'VideoViews' for each country
views_by_country = youtube_df.groupby('Country')['VideoViews'].sum().reset_inde

# Sort values
views_by_country = views_by_country.sort_values(by='VideoViews', ascending=Fals

# Plot
plt.figure(figsize=(12, 8))
plt.bar(views_by_country['Country'], views_by_country['VideoViews'], color='ora
plt.title('Total Video Views by Country')
plt.xlabel('Country')
plt.ylabel('Total Video Views(in Trillions)')
plt.xticks(rotation=45, ha='right')  # Allow us to read labels better
plt.show()
```



From this chart, it is clear that the channels that gain the most views are dominated by 2 countries: Unites States and India. We can also see that there is a large number of views generated by the unspecified country in the group, which makes me think that these missing values could affect the shape of this chart slightly.

In [20]:
```
# Group by 'Country' and sum up the 'VideoViews' and 'Population' for each cou
country_stats = youtube_df.groupby('Country').agg({'VideoViews': 'sum', 'Popul

#Plot
plt.figure(figsize=(12, 8))
plt.scatter(country_stats['Population'], country_stats['VideoViews'], color='s
plt.title('Relationship between Country Population and Video Views')
plt.xlabel('Country Population(In billions)')
plt.ylabel('Total Video Views(In trillions)')
plt.show()
```



It appears that channels that are located in highly populated countries, are able to generate more video views (mainly in United States and India). However, we should note that although China is #2 in world rankings population, it doesn't generate views because YouTube is banned in China(point on the bottom right). Most of the countries are near the bottom left, simply because the distribution is skewed to the right.

#3. Can we build a model that can accurately predict a channel's subscribers count?

The variables that I thought were the most important to build a model are Category, TotalViewCount, Country, ChannelType, HighestYearlyEarnings, Uploads, YearCreated, and Population. I decided to drop the other columns and split the data into training and testing sets. Before we start building a model, we must ensure that we properly encode our categorical variables. We will then start with a Linear Regression model to see how well it performs on our dataset to predict a channel's subscribers count.

In [27]:
```
# Initialize LabelEncoder
label_encoder = preprocessing.LabelEncoder()

# Fit and transform the categorical variables column using .loc
youtube_df.loc[:, 'CategoryEncoded'] = label_encoder.fit_transform(youtube_df[
youtube_df.loc[:, 'CountryEncoded'] = label_encoder.fit_transform(youtube_df['
youtube_df.loc[:, 'ChannelTypeEncoded'] = label_encoder.fit_transform(youtube_
```

In [22]:
```
#Drop variables we won't use
X = youtube_df.drop(columns = ['Rank','Subscribers', 'Youtuber', 'Category', '
y = youtube_df['Subscribers']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando


#Linear Regression Model

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LinearRegression()

model.fit(X_train_scaled, y_train)

predictions1 = model.predict(X_test_scaled)

# RMSE
rmse = sqrt(mean_squared_error(y_test, predictions1))

# Evaluate the model using Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, predictions1)

print(f'Mean Absolute Error on Test Set: {mae:.2f}')
print(f'Root Mean Squared Error on Test Set: {rmse:.2f}')
```

```
Mean Absolute Error on Test Set: 5845131.91
Root Mean Squared Error on Test Set: 8004807.00
```

The Mean Absolute Error(MAE) is the average of the differences(absolute) between the actual and predicted values. It gives an equal weight to all of its error and is not affected as much by outliers. I decided against using RMSE because since our values are extremely large, it would penalize the errors more since we are squaring the values. But for purpose of showing, I will still be including it.

A value of 5845131.91 for our MAE means that on average our model is off by 5,845,131.91 subscribers while trying to predict values for the test set. This does not appear to be a bad model, but let us see if we can improve upon this MAE value by using a random forest model!

In [23]:
```
#Random Forest Model
rf_model = RandomForestRegressor(random_state=50)
rf_model.fit(X_train_scaled, y_train)
predictions2 = rf_model.predict(X_test_scaled)

r2 = r2_score(y_test, predictions2)

# MAE of random forest
mae_rf = mean_absolute_error(y_test, predictions2)

print(f'Mean Absolute Error for Random Forest: {mae_rf:.2f}')
print(f'R-squared for Random Forest: {r2:.4f}')
```

```
Mean Absolute Error for Random Forest: 6115424.24
R-squared for Random Forest: 0.3378
```

A Mean Absolute Error for Random Forest: 6115424.24 means that our model performed worse than linear regression. This could be because we did not tune our hyperparameters properly, and this may have led to overfitting.

I am using GridSearch Cross Validation that will allow us to try out various parameters through cross validation and allow us to select the parameters that give us the lowest Mean Absolute Error.

In [24]:
```
from sklearn.model_selection import GridSearchCV

# Defining the parameter grid
param_grid = {
    'n_estimators': [100, 150, 200, 250, 300, 350, 400],
    'max_depth': [None, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the random forest model
rf_model_tuned = RandomForestRegressor(random_state=50)


grid_search = GridSearchCV(rf_model_tuned, param_grid, cv=5, scoring='neg_mean_
grid_search.fit(X_train_scaled, y_train)

# Getting the best parameters
best_params = grid_search.best_params_
best_rf_model = grid_search.best_estimator_

# Make predictions on test set
predictions = best_rf_model.predict(X_test_scaled)


mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

# Print evaluation metrics
print(f'Best Parameters: {best_params}')
print(f'Mean Absolute Error on Test Set: {mae:.2f}')
print(f'Mean Squared Error on Test Set: {mse:.2f}')
print(f'R-squared on Test Set: {r2:.4f}')
```

```
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split':
10, 'n_estimators': 250}
Mean Absolute Error on Test Set: 6105461.83
Mean Squared Error on Test Set: 77763396028984.11
R-squared on Test Set: 0.3125
```

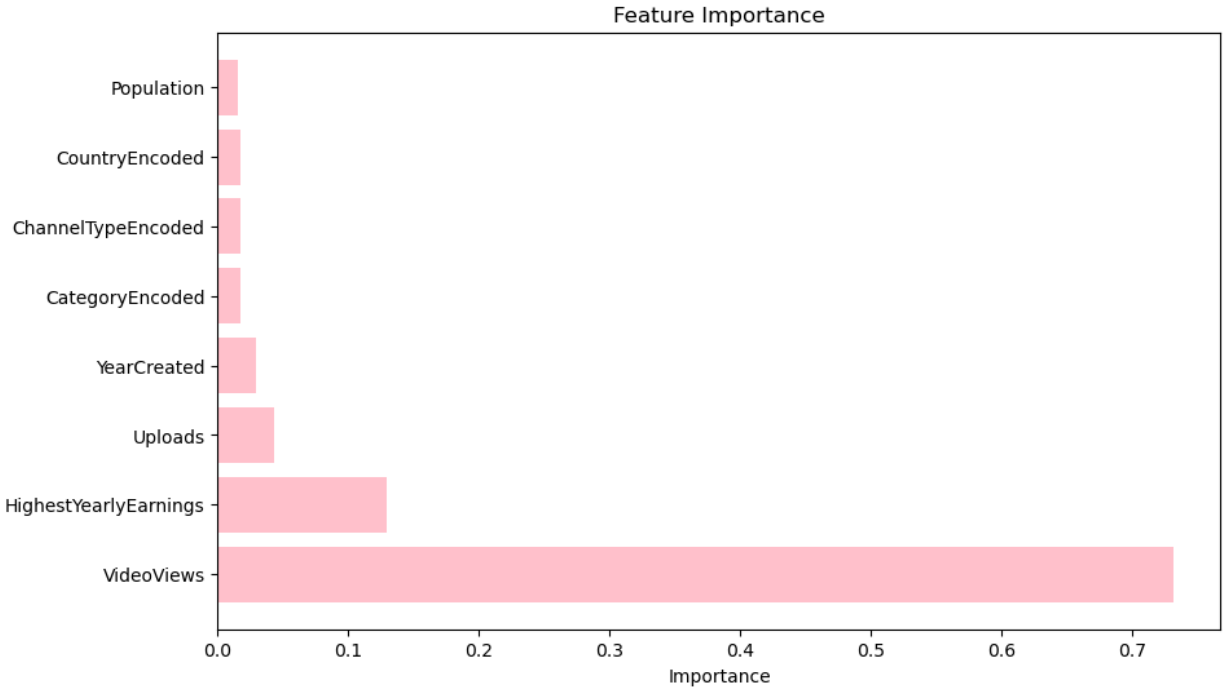
Upon performing hyperparameter tuning to find the best parameters for our random forest model, I was able to slightly improve the model's accuracy on the test set. The Mean Absolute Error on Test Set is 6105461.83 which means that on average our model is off by 6,105,461.83 subscribers while trying to predict values for the test set. The R-squared value on Test Set is 0.3125, which means that the model explain only 31.25% of the variance on

the variable:subscribers. The reason that our model may not have performed too well is there could be external factors outside of our data set that are affecting the number of subscribers a channel has. Random Forests are good for complex/non-linear relationships, so it could also be the case that our problem consists of a linear relationship.

Let us now see how important each feature is.

In [25]:
```
# Get feature importances
feature_importance = rf_model.feature_importances_

feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': featu

#sort by descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance', asc

print(feature_importance_df)

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance']
plt.xlabel('Importance')
plt.title('Feature Importance')
plt.show()
```

```
                Feature  Importance
0            VideoViews    0.731113
2  HighestYearlyEarnings    0.129751
1               Uploads    0.042842
3           YearCreated    0.029115
5       CategoryEncoded    0.017540
7    ChannelTypeEncoded    0.017120
6        CountryEncoded    0.016951
4            Population    0.015567
```



It appears that the most important features in our dataset are VideoViews and HighestYearlyEarnings, while all of the other variables appear to have little to none influence on our dataset.

Presenting My Findings

1

Some speculate that in order for a YouTube channel to be successful, one needs to continuously upload videos either daily or at least a few times per week. This was one of the main reasons why I wanted to explore to see if the number of uploads amongst those in the top 1000 accounts were correlated to their subscriber or video view count. After looking at the scatterplots it seems that even if a channel uploads a lot of videos, they may not have a lot of subscribers. I thought that this was interesting because I thought that YouTube's algorithm that helps YouTubers reach out to more people would allow these channels to get their videos across to a wider audience and that if they had more videos, people would be more inclined to subscribe. One possible limitation I experienced with tackling this question was that this dataset was only limited to the most popular youtube channels across the entire world and does not include the average YouTube channel. So these results should only be limited to those that are amongst the top YouTube ranked channels.

We were also able to notice that subscribers and video view counts have a strong positive correlation. This makes sense because those who have more subscribers would have more people regularly watching their content so they would be able to have more views consistently on their videos. We were also able to see that highest yearly earnings column is correlated to higher subscribers which can indicate that those who have higher subscribers make more money because they are able to generate more views for their videos. YouTube's monetization policy requires at least 500 subscribers, 3 public uploads in the last 90 days, 3,000 public watch hours in the previous 12 months or 3 million public YouTube Shorts views in the last 90 days. This means that those who want to join YouTube's creator program to earn money must be consistent in uploading their videos and need to generate enough views.

Some problems that I experienced with tackling this question was that this dataset was only limited to the most popular youtube channels across the entire world and does not include the average YouTube channel. So these results should only be limited to those that are amongst the top YouTube ranked channels. Another limitation that I had was that most of the data was skewed to the right, which meant that the top few channels that had a lot more subscribers than others were heavily influencing my results in the plot. I think that my results would be even more interesting if I was able to look further into a larger dataset that included average and new YouTube channels.

2

We looked at the relationship between a channel's country of origin and the total number of views earlier and found that most of the views were from those in the United States and India, with a large number in an unspecified group. It turns out that although YouTube was founded in the United States, India has the largest user population of the platform. As of October 2023, India is the country with the largest YouTube audience by far, with approximately 462 million users. The United States followed, with around 239 million YouTube viewers. This perfectly aligns with my results because those with channels from these countries are able to generate the most views because they have the most users using the platform.

Since both of these countries also have a large population (India: 1st and USA: 3rd), it would safe to assume that these are the main factors contributing to these large number of views in their respective countries.

After seeing a large number of channels labeled in the Unspecified group (approximately 10%), it led me to believe that some of this data was missing from the dataset which could have influenced by bar chart. I still believe that the shape/curve of the graph would remain about the same, but it still worth noting that there was a large number of missing values in the Country column.

3

After seeing that both linear regression and random forest models both performed poorly on the dataset, it leads me to believe that there could be external factors that need to be accounted for in order to be able to predict a channel's subscribers. I also found that a lot of variables that I had originally thought were important to a channel's subscriber counts such as Uploads, Category, Country, and Population of Country were actually not important at all. This means that there are most definitely other factors that are not quantifiable and could be affected by user's preferences.

For instance, YouTube is constantly changing the trending topics that are affected by global events. YouTubers will be affected by this because they will need to constantly adapt to what is trending and create new appealing content in line with trends. A big reason why YouTubers have high subscriber counts is also because they are actively engaging with their subscribers and community by replying to comments and doing specific videos for their fans. These factors are challenging to quantify as they are extremely unpredictable and always changing(dynamic). Predictive models may not be able to account for these factors because YouTube consists of non-quantifiable elements that affect user's preferences.

Conclusion

In conclusion, the Global YouTube Statistics 2023 Dataset has provided valuable insights into the dynamics of top YouTube channels worldwide. The analysis covered the various factors that influenced subscriber counts such as video view count and highest yearly earnings which revealed to us that there are potentially external factors that affect a channel's subscriber count. With an initial belief that if channels were to simply upload more videos they would be able to generate more views and gain subscribers, I was proved wrong. The strong positive correlation between subscribers and video views showed us how important it is for subscribers to engage with their community to improve viewer engagement. With the dataset's limitations on missing values and skewed distributions of several columns, we now know that we must look for a more comprehensive dataset that includes average youtube channels information. Furthermore, by attempting to predict subscriber counts using linear regression and random forest models we ecountered difficulties which suggested that there was an influence of unquantifiable external factors that affects a channels ability to gain new subscribers.