|  |  |
| --- | --- |
| Project Title | **Analyzing YouTube Channel Statistics** |
| Languages | Machine learning, python, MYSQL, Excel |
| Tools | Visual Studio code / Jupyter notebook |
| Domain | Data Analyst |
| Project Difficulties level | Intermediate |

Dataset: Dataset is available in the given link. You can download it at your convenience.

[Click here to download data set](https://docs.google.com/spreadsheets/d/1rSjGKcyl3l2ZPd2KMmSmBxV22mXRG5ab/edit?usp=drivesdk&ouid=103470733098949669049&rtpof=true&sd=true)

**About Dataset**

This dataset offers a comprehensive overview of the top YouTube channels, focusing on key performance indicators that contribute to channel popularity, audience growth, and content strategy. It captures essential metrics such as subscriber count, total video views, number of videos, content category, and the year the channel was started. By profiling 1000 of the most subscribed YouTube channels, the dataset provides insights into how creators across genres have scaled their reach and engagement.

**Columns include:**

* **Rank:** Global rank based on subscribers.
* **Youtuber**: Name of the channel.
* **Subscribers**: Total number of subscribers.
* **Video** **Views**: Cumulative views across all videos.
* **Video** **Count**: Total number of uploaded videos.
* **Category**: Primary content genre (e.g., Music, Entertainment, Education).
* **Started**: Year the channel was create

**Major Machine Learning Project: Analyzing YouTube Channel Statistics**

This project aims to analyze YouTube channel performance by leveraging extensive metrics and using Machine Learning techniques to uncover patterns, trends, and actionable insights. We'll focus on **Exploratory Data Analysis (EDA)**, **data visualization**, and developing a **predictive model** to estimate subscribers based on the provided dataset. Ultimately, the project aims to provide actionable insights that content creators can use to optimize their growth strategies and maximize channel performance.

# Step-by-Step Workflow

1. **Import Libraries**

python code

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.ticker as mticker

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

# Load and Explore the Dataset

python code

# Load the dataset

data = pd.read\_csv('D:/Notes/Python/Unified Mentor/Top Youtubers Dataset.csv',encoding='latin1')

# Preview the dataset data.head()

# Display basic information about the dataset print(data.info())

# Display Index print(data.index)

# Display Columns

print(data.columns)

# Check shape

print(data.shape)

# Check for null values print(data.isnull().sum())

# Data Cleaning Handle Missing Values:

python code

# Drop null values, Drop rows with missing values(for simplicity)

data.dropna(inplace=True)

**● Rename Columns:**

python code

# Rename the columns for your convenience

data = data.rename(columns={'Youtuber':'Channel', 'Video Views':'Views', 'Video Count':'Uploads', 'Started':'Created On'})

1. Exploratory Data Analysis (EDA) Analyze relationships:

python code

# Pairplot to visualize relationships sns.pairplot(data[['Subscribers', 'Uploads', 'Views']])

plt.show()

● Correlation Heatmap:

python code

corr\_matrix = data.corr(numeric\_only=True)

plt.figure(figsize=(13, 8))

sns.heatmap(corr\_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5, square=True, cbar\_kws={'label': 'Correlation Coefficient'})

plt.title("Correlation Heatmap of Numeric Features",fontsize=16)

plt.show()

● Top Performers by Views and Subscribers:

python code

# Create top\_Views

top\_Views = data.sort\_values(by='Views',

ascending=False).head(10).reset\_index(drop=True)

# Create top\_Subscribers

top\_Subscribers = data.sort\_values(by='Subscribers', ascending=False).head(10).reset\_index(drop=True)

# Feature Engineering

Create new features:

python code

# Create subscribers by year

subscribers\_by\_year = data.groupby('Created On')['Subscribers'].mean().reset\_index()

# Create One-hot encoding

data\_encoded = pd.get\_dummies(data[['Category']])

**6.Data Visualization Subscribers Distribution:**

python code

plt.figure(figsize=(15, 6))

sns.histplot(data['Subscribers'], bins=50, kde=True,

color='blue', edgecolor='black', alpha=0.6)

sns.kdeplot(data['Subscribers'])

plt.title("Distribution of Subscribers", size=16, color='blue')

plt.xlabel("Subscribers(in million)", size=12)

plt.ylabel("Number of Channels", size=12)

plt.grid(True, linestyle='--', alpha=0.4)

plt.tight\_layout()

plt.show()

● Year Channel Was Created Distribution:

python code

plt.figure(figsize=(15,6))

sns.histplot(data['Created On'], bins=50, kde=True, alpha=0.6, color='skyblue', edgecolor= 'black')

plt.title("Distribution of Year Channel Was Created", size=16)

plt.xlabel("Year Channel Was Created", size=12)

plt.ylabel("Number of Channels", size=12)

plt.grid(True, linestyle='--',alpha=0.4)

plt.tight\_layout()

plt.show()

● Subscribers vs Views:

python code

plt.figure(figsize=(15,7))

sns.scatterplot(x='Subscribers', y='Views', data=data,

hue='Category', alpha=0.7)

plt.title('Subscribers vs. Views', size=16, color='orange')

plt.xlabel('Subscribers', size=12)

plt.ylabel('Views', size=12)

plt.grid(True, alpha=0.6, linestyle='--')

plt.legend(loc='upper left', bbox\_to\_anchor=(1,1))

plt.tight\_layout()

plt.show()

● Views vs Channels:

python code

plt.figure(figsize=(15, 6))

sns.barplot(x='Views', y='Channel', data=top\_Views, palette='magma')

plt.title('Top 10 YouTube Channels by Video Views', fontsize=16)

plt.xlabel('Views', fontsize=12)

plt.ylabel('Channels', fontsize=12)

# Format x-axis to billions

plt.gca().xaxis.set\_major\_formatter(mticker.FuncFormatter(lambda x, \_: f'{x/1e9:.1f}B'))

# Add annotations

for i, v in enumerate(top\_Views['Views']):

plt.text(v + 1e9, i, f'{v/1e9:.1f}B', va='center', fontsize=10)

plt.tight\_layout()

plt.show()

● Category vs Channels:

python code

plt.figure(figsize=(15,7))

sns.countplot(x='Category', data=data, order=data['Category'].value\_counts().index, palette='coolwarm')

plt.title('Distribution of Channels by Category', size=16, color='blue')

plt.xlabel('Category', size=12)

plt.ylabel('Number of Channels', size=12)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

● Average Subscribers vs Over Years:

python code

plt.figure(figsize=(15,7))

sns.lineplot(x='Created On', y='Subscribers',

data=subscribers\_by\_year, marker='o')

plt.title('Average Subscribers Over Years', size=16, color='Navy')

plt.xlabel('Year Created', size=12)

plt.ylabel('Average Subscribers', size=12)

plt.grid(True, alpha=0.4, linestyle='--')

plt.tight\_layout()

plt.show()

# Predictive Model: Estimate Revenue Prepare Data:

python code

# Select features and target

features = ['Views', 'Uploads', 'Created On']

target = 'Subscribers'

X = data[features]

y = data[target]

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

● Train Random Forest Regressor:

python code

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = model.predict(X\_test)

● Evaluate the Model:

python code

# Calculate performance metrics

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

r2 = r2\_score(y\_test, y\_pred)

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"R² Score: {r2}")

# Insights and Recommendations

Use visualizations and feature importance to derive insights: python

code

# Feature Importance

importances = model.feature\_importances\_

feature\_importance\_df = pd.DataFrame({'Feature': features, 'Importance': importances})

feature\_importance\_df = feature\_importance\_df.sort\_values(by='Importance',ascending = False)

# Plotting feature importance

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

sns.barplot(x='Importance', y='Feature', data=feature\_importance\_df)

plt.title("Feature Importance for Predicting Subscribers")

plt.show()

1. Deployment and Presentation
   * Summarize findings:
     + Highlight top Subscribers Channels (e.g., Views, Uploads).
     + Identify performing areas (e.g., Entertainment or Music).

Export model:

python code

import joblib

joblib.dump(model, 'youtube\_subscribers\_predictor.pkl

Sample code with output

In the world of YouTube, where number of views, uploads and category are matter to increase the subscribers. This dataset offers a treasure trove of insights into YouTube channel analytics. Let's dive into the data and see what stories it has to tell.

In [1]:

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import matplotlib.ticker as mticker

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

In [2]:

*# Load the dataset*

data = pd.read\_csv('D:/Notes/Python/Unified Mentor/Top

Youtubers Dataset.csv',encoding ='latin1')

data.head()

Out [2]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rank** | **Youtuber** | **Subscribers** | **Video Views** | **Video Count** | **Category** | **Started** |
| 1 | MrBeast | 284000000 | 52402898889 | 803 | Entertainment | 2012 |
| 2 | T-Series | 268000000 | 259000000000.0 | 21237 | Music | 2006 |
| 3 | YouTube Movies | 181000000 | 0 | 0 | Film & Animation | 2015 |
| 4 | Cocomelon - Nursery Rhymes | 177000000 | 183000000000.0 | 1188 | Education | 2006 |
| 5 | SET India | 174000000 | 165000000000.0 | 139720 | Shows | 2006 |

Data Overview

Let's take a look at the basic information about the dataset to understand its structure and contents.

In [3]:

data.info()

Out [3]:

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

RangeIndex: 1000 entries, 0 to 999

Data columns (total 7 columns):

# Column Non-Null Count Dtype

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

0 Rank 1000 non-null int64

1 Youtuber 1000 non-null object

2 Subscribers 1000 non-null int64

3 Video Views 1000 non-null float64

4 Video Count 1000 non-null int64

5 Category 957 non-null object

6 Started 1000 non-null int64

dtypes: float64(1), int64(4), object(2)

memory usage: 54.8+ KB

Data Cleaning and Preprocessing

Before diving into analysis, let's ensure the data is clean and ready for exploration.

In [4]:

*# Check for missing values*

data.isnull().sum()

Out [4]:

Rank 0

Youtuber 0

Subscribers 0

Video Views 0

Video Count 0

Category 43

Started 0

dtype: int6

In [5]:

*# Renaming the columns*

data = data.rename(columns = {'Youtuber': 'Channel',

'Video Views': 'Views', 'Video Count': 'Uploads', 'Started': 'Created On'})

**Exploratory Data Analysis**

Let's explore the data to uncover patterns and insights.

In [6]:

*#* **Pairplot to visualize relationships**

sns.pairplot(data[['Subscribers', 'Uploads', 'Views']])

plt.show()

Out [6]:

A group of blue dots

AI-generated content may be incorrect.

In [7]:

*#* Correlation Heatmap:

plt.figure(figsize = (13, 8))

*# Compute the correlation matrix*

corr\_matrix = data.corr(numeric\_only = True)

*# Plot the heatmap*

sns.heatmap(corr\_matrix, annot = True, fmt = ".2f", cmap = "coolwarm", linewidths = 0.5, square = True, cbar\_kws = {'label'

: 'Correlation Coefficient'})

plt.title("Correlation Heatmap of Numeric Features", fontsize = 16)

plt.show()

Out [7]:

A screenshot of a graph

AI-generated content may be incorrect.

In [8]:

*# Top Performers by Views and Subscribers:*

top\_Views = data.sort\_values(by = 'Views',

ascending = False).head(10).reset\_index(drop = True)

top\_Subscribers = data.sort\_values(by = 'Subscribers', ascending=False).head(10).reset\_index(drop = True)

In [9]:

**# Subscribers Distribution:**

plt.figure(figsize =(15, 6))

sns.histplot(data['Subscribers'], bins = 50, kde = True, color

= 'blue', edgecolor = 'black', alpha = 0.6)

sns.kdeplot(data['Subscribers'])

plt.title("Distribution of Subscribers", size = 16, color='blue')

plt.xlabel("Subscribers(in million)", size = 12)

plt.ylabel("Number of Channels", size = 12)

plt.grid(True, linestyle='--', alpha = 0.4)

plt.tight\_layout()

plt.show()

Out [9]:

A graph with numbers and lines

AI-generated content may be incorrect.

In [10]:

**# Distribution of Year Channel Was Created:**

plt.figure(figsize = (15,6))

sns.histplot(data['Created On'], bins = 50, kde = True, alpha = 0.6, color = 'skyblue', edgecolor = 'black')

plt.title("Distribution of Year Channel Was Created", size = 16)

plt.xlabel("Year Channel Was Created", size = 12)

plt.ylabel("Number of Channels", size = 12)

plt.grid(True, linestyle = '--',alpha = 0.4)

plt.tight\_layout()

plt.show()

Out [10]:

A graph with blue lines

AI-generated content may be incorrect.

In [11]:

# scatterplot of Subscribers and Views:

plt.figure(figsize = (15,7))

sns.scatterplot(x='Subscribers', y ='Views', data = data, hue ='Category', alpha = 0.7)

plt.title('Subscribers vs. Views', size = 16, color = 'orange')

plt.xlabel('Subscribers', size = 12)

plt.ylabel('Views', size = 12)

plt.grid(True, alpha = 0.6, linestyle = '--')

plt.legend(loc = 'upper left', bbox\_to\_anchor = (1,1))

plt.tight\_layout()

plt.show()

Out [11]:

A screen shot of a graph

AI-generated content may be incorrect.

In [12]:

# Views vs Channels

plt.figure(figsize = (15, 6))

sns.barplot(x = 'Views', y = 'Channel', data = top\_Views, palette = 'magma')

plt.title('Top 10 YouTube Channels by Video Views', fontsize =

16)

plt.xlabel('Views', fontsize = 12)

plt.ylabel('Channels', fontsize = 12)

# Format x-axis to billions

plt.gca().xaxis.set\_major\_formatter(mticker.FuncFormatter(lambda x, \_: f'{x/1e9:.1f}B'))

# Add annotations

for i, v in enumerate(top\_Views['Views']):

    plt.text(v + 1e9, i, f'{v/1e9:.1f}B', va = 'center', fontsize = 10)

plt.tight\_layout()

plt.show()

Out [12]:

A graph of a bar chart

AI-generated content may be incorrect.

In [13]:

# plot countplot on Channels vs Category

plt.figure(figsize = (15,7))

sns.countplot(x = 'Category', data = data, order = data['Category'].value\_counts().index, palette = 'coolwarm')

plt.title('Distribution of Channels by Category', size = 16, color = 'blue')

plt.xlabel('Category', size = 12)

plt.ylabel('Number of Channels', size = 12)

plt.xticks(rotation = 45)

plt.tight\_layout()

plt.show()

Out [13]:

A graph with different colored squares

AI-generated content may be incorrect.

in [14]:

# Lineplot of Created On vs Subscribers

plt.figure(figsize = (15,7))

sns.lineplot(x = 'Created On', y = 'Subscribers', data = subscribers\_by\_year, marker =

'o')

plt.title('Average Subscribers Over Years', size = 16, color = 'Navy')

plt.xlabel('Year Created', size = 12)

plt.ylabel('Average Subscribers', size = 12)

plt.grid(True, alpha = 0.4, linestyle = '--')

plt.tight\_layout()

plt.show()

Out [14]:

A graph with a line going up

AI-generated content may be incorrect.

Predictive Modeling

In [15]:

*# Define features and target variable*

features = ['Views', 'Uploads', 'Created On']

target = 'Subscribers'

X = data[features]

y = data[target]

*# Split the data into training and testing sets*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)

*# Initialize and train the model*

model = RandomForestRegressor(n\_estimators = 100, random\_state

= 42)

model.fit(X\_train, y\_train)

*# Make predictions*

y\_pred = model.predict(X\_test)

*# Calculate the prediction accuracy*

mse = mean\_squared\_error(y\_test, y\_pred) rmse = np.sqrt(mse)

rmse

Out[15]:

0.3559392144882

linkcode Discussion

In this notebook, we explored a comprehensive YouTube channel performance dataset. We visualized key metrics, examined correlations, and built a predictive model for estimating revenue. The Random Forest model provided a reasonable prediction accuracy, but there's always room for improvement. Future analysis could explore feature engineering, hyperparameter tuning, or even different modeling approaches to enhance prediction performance. If you found this analysis insightful, please consider upvoting this notebook.