Youtube Adview Prediction

Dr C K Gomathy, A. V. S. Hrushikesh, A. Vishnu Vardhan.
Sri Chandrasekharendra SaraswathiViswa Mahavidyalaya , Kanchipuram

Abstract:

Predicting Web content popularity is an important task for supporting the design and evaluation of a wide range of systems, from targeted advertising to effective search and recommendation services. We here present two simple models for predicting the popularity of Youtube content based on historical information given by early popularity measures.Youtube advertisers or the youtube will pay it's content creators based on adviews and clicks for the goods and services being marketed or the content they upload. They want to estimate the adview based on other metrics like comments, likes, dis-likes etc. In this project we are going to train various regression models and choose the best one to predict the number of adviews a video can get based upon the metrics like views, likes, dis-likes etc. . The data needs to be refined and cleaned before feeding in the algorithms for better results.

Keywords: Adview Techniques, Web Content Trends, Youtube, YouTube prediction

I. Introduction:

Over the past 5 years YouTube has paid out more than $5 billion to YouTube content creators. Popular YouTuber PewDiePie made $5 million in 2016 from YouTube alone, not including sponsorships, endorsements and other deals outside of YouTube. With more and more companies turning to YouTube influencers to capture the millennial audience, getting people to watch your videos on YouTube is becoming increasingly lucrative.

Our goal is to create a model that can help youtube content creators, predict the number of adviews for their next video. Content on Youtube covers a broad range of genres such as comedy, tech, sports, fashion, gaming and fitness.

A person looking at related videos suggested by YouTube will first see the title and the likes it got previously. If more potential views can be generated with specific titles and thumbnails, a YouTuber could use this information to generate the maximum potential views with video content they worked hard on. Therefore, our goal was to create a model using non-video features to predict the view count that youtube content creators can use to help grow their channel. We will build a machine learning regression to predict youtube adview count based on other youtube metrics.

II. Literature Survey/Existing System:

1. Zhou et al. studied the impact of YouTube recommendation system on video views and concluded that there is a strong correlation between view count of a video and average view count of its top referred video by recommendation system (Acm.org, 2010).
2. Also, Davidson et al. studied recommendation system through CTR (click through rate) of videos on home page. They conclude that recommendation by YouTube account for 60% of all video clicks on YouTube homepage (Davidson et al., 2010).
3. Now, a few approaches have been made towards YouTube trending videos research such as Prabha et. al. discussed predicting the popularity of trending videos in YouTube using sentiment analysis. They had chosen the NLP path to predict the popularity of trending videos. After discussing various classifiers such as naïve Bayes and KNN for building their model, Prabha et. al. proposed an algorithm using SVM to predict trending videos popularity and concludes that their model can help increase accuracy of such predictive analysis (Prabha, G.M. et al., 2019).

III. Proposed System:

To build a machine learning regression to predict youtube adview count based on other youtube metrics.

Attribute Information:

1. 'vidid' : Unique Identification ID for each video
2. 'adview' : The number of adviews for each video created by youtuber
3. 'views' : The number of unique views for each video
4. 'likes' : The number of likes for each video
5. 'dislikes' : The number of likes for each video
6. 'comment' : The number of unique comments for each video
7. 'published' : The data of uploading the video
8. 'duration' : The duration of the video (in min. and seconds)
9. 'category' : Category niche of each of the video

IV. Methodology:

1. Import the datasets and required libraries then, check shape and datatype.
2. Visualise the dataset using plotting using heatmaps and plots or any other data visualising techniques available in pandas and seaborn. You can study data distributions for each attribute as well.
3. Clean the dataset by removing missing values and other things. Cleaning data is an import task to get efficient model.
4. Transform attributes into numerical values and other necessary transformations
5. Normalise your data and split the data into training, validation and test set in the appropriate ratio.
6. Use Decision Tree Regressor and Random Forest Regressors.
7. Build an artificial neural network and train it with different layers and hyperparameters.
8. Pick the best model based on error as well as generalisation.
9. Save your model and predict on the test set.

V. Design and Development:

1. **Import the datasets and libraries, check shape and datatype.**

You basically import preinstalled python libraries or packages like numpy, pandas, matplotlib and seaborn for data cleaning and visualisation. Scikitlearn and Keras are imported for machine learning models and neural networks. Using the pandas library we can simply import the dataset in csv format as a pandas dataframe. We can then explore the dataset and check the number of features and samples in the data.

```
import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.pyplot as plt
# Importing data

path = "" # Put path of your folder of your data if it's not in the same folder
data_train = pd.read_csv(path + "train.csv")

data_train.head()

data_train.shape
# (14999, 9)
```

2. **Visualise the dataset using plotting using heatmaps and plots. You can study data distributions for each attribute as well.**

Matplotlib and seaborn libraries can be used for plotting. We can plot for each of the features and visually see the distribution of the data with respect to that feature. We can also use it to spot any outliers whatsoever which will help our model to train and learn better. We can plot a heatmap using the seaborn library where

we can visualise the correlations of different features with respect to each other. It helps to see how independent are the features because we may want to remove the features
which may not add any value to our model.

```
# Visualization
    # Individual Plots
plt.hist(data_train["category"])
plt.show()
plt.plot(data_train["adview"])
plt.show()

    # Remove videos with adview greater than 2000000 as outlier
data_train = data_train[data_train["adview"] <2000000]
```

```
   # Heatmap
import seaborn as sns

f, ax = plt.subplots(figsize=(10, 8))
corr = data_train.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
           square=True, ax=ax,annot=True)
plt.show()
```

3. **Clean the dataset by removing missing values and other things.**

Cleaning the dataset is one of the most essential tasks while solving a machine learning problem. Here in this problem we can remove 'F' and other missing values that might hurt the performance of our model.

```
# Removing character "F" present in data
data_train=data_train[data_train.views!='F']
data_train=data_train[data_train.likes!='F']
data_train=data_train[data_train.dislikes!='F']
data_train=data_train[data_train.comment!='F']

data_train.head()

# Assigning each category a number for Category feature
category={'A': 1,'B':2,'C':3,'D':4,'E':5,'F':6,'G':7,'H':8}
data_train["category"]=data_train["category"].map(category)
data_train.head()
```

4. **Transform attributes into numerical values and other necessary transformations**

Machine Learning models need data to be converted to numerical form at the end. We have categorical data and data in other formats which may want to convert. We will use label encoder and other date & time

functions for that task. This part is quite open ended and also known as feature engineering. You can transform original features into new ones which might give better results.

```
# Convert values to integers for views, likes, comments, dislikes and adview
data_train["views"] = pd.to_numeric(data_train["views"])
data_train["comment"] = pd.to_numeric(data_train["comment"])
data_train["likes"] = pd.to_numeric(data_train["likes"])
data_train["dislikes"] = pd.to_numeric(data_train["dislikes"])
data_train["adview"]=pd.to_numeric(data_train["adview"])

column_vidid=data_train['vidid']

# Endoding features like Category, Duration, Vidid
from sklearn.preprocessing import LabelEncoder
data_train['duration']=LabelEncoder().fit_transform(data_train['duration'])
data_train['vidid']=LabelEncoder().fit_transform(data_train['vidid'])
data_train['published']=LabelEncoder().fit_transform(data_train['published'])

data_train.head()

# Convert Time_in_sec for duration
import datetime
import time
```

5. **Normalise your data and split the data into training, validation and test set in the appropriate ratio.**

We need to split the data into training and test data. We use training data to learn patterns in the data and then check if it generalises well on unseen data. The split percentage can be varied and is generally on the amount of data we have. For a relatively small dataset like this, the split percentage should be high 80 : 20 rather than 99 : 1 with respect to train : test). Normalisation is done to ensure all the features all weighted appropriately in the training stage. Just because some features have high scale should not translate to having higher influence on the model. Normalisation can be done using Standard Scalar or MinMax Scalar among others. In this particular problem, MinMax Scalar has been used which basically transforms each variable in the range of 0 to 1.

```
# Split Data
Y_train  = pd.DataFrame(data = data_train.iloc[:, 1].values, columns = ['target'])
data_train=data_train.drop(["adview"],axis=1)
data_train=data_train.drop(["vidid"],axis=1)
data_train.head()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_train, Y_train, test_size=0.2, random_state=42)

X_train.shape

# Normalise Data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.fit_transform(X_test)

X_train.mean()
```

6. **Use Decision Tree Regressor and Random Forest Regressors.**

Another class of machine learning algorithms include decision trees and random forests. We can import these models from sklearn.tree and then again use the .fit function. We need to give appropriate hyper parameters for them. These are something we can experiment with to achieve better results.

```
# Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
decision_tree = DecisionTreeRegressor()
decision_tree.fit(X_train, y_train)
print_error(X_test,y_test, decision_tree)

# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
n_estimators = 200
max_depth = 25
min_samples_split=15
min_samples_leaf=2
random_forest = RandomForestRegressor(n_estimators = n_estimators, max_depth = max_depth, min_samples_split=min_samples_spl
random_forest.fit(X_train,y_train)
print_error(X_test,y_test, random_forest)
```

The best result was given by the Decision Tree Reggressor.

VI. Results:

For Decision Tree Regressor:

Mean Absolute Error: 2743.0823087431695

Mean Squared Error: 892481316.1048497

Root Mean Squared Error: 29874.42578703145

For Artificial Neural Network:

Total params: 97

Trainable params: 97

Non-trainable params: 0

Mean Absolute Error: 3308.1026117736524

Mean Squared Error: 829841746.1640484

Root Mean Squared Error: 28806.97391542625

VII. Conclusion:

Since Decision Tree Regressor is having least Mean Absolute Error as compared to Neural Network , we will pickle and save that decision tree regressor model to predict.

Popularity is often considered a random act. Analyzed YouTube's adview forecasting aspect to understand the randomness of a video's virality and channel monetory for academic and business perspective. This implemented Decision tree regressor and Artificial Neural Network in 'Comparative analysis of Machine Learning algorithms for YouTube adview prediction by analysing YouTube's statistics data'. Study also concludes the scope of better classification results by performing the existing methodology on a better supervised dataset.

VII. References:

- G. Gursun, M. Crovella, and I. Matta, "Describing and forecasting € video access patterns," in Proc. IEEE INFOCOM, 2011, pp. 16–20
- H. Pinto, J. Almeida, and M. Gonc¸alves, "Using early view pat-terns to predict the popularity of YouTube videos," in Proc. 6th ACM Int. Conf. Web Search Data Mining, 2013, pp. 365–374.

- C. Richier, E. Altman, R. Elazouzi, T. Jimenez, G. Linares, and Y. Portilla, "Bio-inspired models for characterizing YouTube viewcout," in Proc.IEEE/ACM Int. Conf. Advances Social Netw. Anal. Mining, 2014, pp. 297–305.
- C. Richier, R. Elazouzi, T. Jimenez, E. Altman, and G. Linares,"Forecasting online contents' popularity," arXiv:1506.00178, 2015.
- [5] A. Zhang, "Judging YouTube by its covers," Dept. Comput.Sci. Eng., Univ. California, San Diego, 2015. [Online]. Available: http://cseweb.ucsd.edu/ jmcauley/cse255/reports/wi15/Angel%20Zhang.pdf

Profile:

1. Mr. A. V. S. Hrushikesh, Student, B.E. Computer Science and Engineering, Sri Chandrasekharendra SaraswathiViswa Mahavidyalaya deemed to be university, Enathur, Kanchipuram, India. His Area of Big Data Analytics,Interest Internet of things.

2. Mr. A. Vishnu Vardhan Student, B.E. Computer Science and Engineering, Sri Chandrasekharendra SaraswathiViswa Mahavidyalaya deemed to be university, Enathur, Kanchipuram, India. Her Area of Big Data Analytics Internet of things.

3. Dr.C.K.Gomathy is Assistant Professor in Computer Science and Engineering at Sri Chandrasekharendra SaraswathiViswa Mahavidyalaya deemed to be university, Enathur, Kanchipuram, India. Her area of interest is Software Engineering, Big Data Analytics Web Services, Knowledge Management and IOT.