Solution : Youtube Adview Prediction

Steps and Tasks

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, pandas, matplotlib and seaborn
   - [ ] Import dataset using pandas.csv() method
   - [ ] Use data.shape etc.

```
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.

   - [ ] Plot distributions
   - [ ] Plot heatmap of correlation values
   - [ ] Remove outliers

```
# 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.

   - [ ] Remove missing values
   - [ ] Remove any other unimportant feature or piece of data

```
# 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 categorical data using label encoder
   - [ ] Convert into numerical data
   - [ ] Feature selection

```
# 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
```

```
def checki(x):
    y = x[2:]
    h = ''
    m = ''
    s = ''
    mm = ''
    P = ['H','M','S']
    for i in y:
        if i not in P:
            mm+=i
        else:
            if(i=="H"):
                h = mm
                mm = ''
            elif(i == "M"):
                m = mm
                mm = ''
            else:
                s = mm
                mm = ''
    if(h==''):
        h = '00'
    if(m == ''):
        m = '00'
    if(s==''):
        s='00'
    bp = h+':'+m+':'+s
    return bp
train=pd.read_csv("train.csv")
mp = pd.read_csv(path + "train.csv")["duration"]
time  = mp.apply(checki)

def func_sec(time_string):
    h, m, s = time_string.split(':')
    return int(h) * 3600 + int(m) * 60 + int(s)

time1=time.apply(func_sec)

data_train["duration"]=time1
data_train.head()
```