Data Description

The file train.csv contains metrics and other details of about 15000 youtube videos. The metrics include number of views,likes,dislikes,comments and apart from that published date,duration and category are also included. The train.csv file also contains the metric number of adviews which is our target variable for prediction

Importing Libraries

In [ ]:

```python
import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Importing data

In [3]:

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

In [4]:

```python
data_train.head()
```

Out[4]:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	9/14/2016	PT7M37S	F
1	VID_14135	2	1707	56	2	6	10/1/2016	PT9M30S	D
2	VID_2187	1	2023	25	0	2	7/2/2016	PT2M16S	C
3	VID_23096	6	620860	777	161	153	7/27/2016	PT4M22S	H
4	VID_10175	1	666	1	0	0	6/29/2016	PT31S	D

In [5]:

```python
data_train.shape
```

Out[5]:

```
(14999, 9)
```

Assigning each category a number for Category feature

In [7]:

```python
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()
```

Out[7]:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	9/14/2016	PT7M37S	6
1	VID_14135	2	1707	56	2	6	10/1/2016	PT9M30S	4
2	VID_2187	1	2023	25	0	2	7/2/2016	PT2M16S	3
3	VID_23096	6	620860	777	161	153	7/27/2016	PT4M22S	8
4	VID_10175	1	666	1	0	0	6/29/2016	PT31S	4

Removing character "F" present in data

In [8]:

```python
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']
```

Convert values to integers for views, likes, dislikes, comments and adview

In [9]:

```python
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"])
```

In [10]:

```python
column_vidid = data_train['vidid']
```

Encoding features like category, Duration, Vidid

In [11]:

```python
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'])
```

In [12]:

```python
data_train.head()
```

Out[12]:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	5912	40	1031602	8523	363	1095	2235	2925	6
1	2741	2	1707	56	2	6	207	3040	4
2	8138	1	2023	25	0	2	1905	1863	3
3	9005	6	620860	777	161	153	1952	2546	8
4	122	1	666	1	0	0	1783	1963	4

Convert Time_in_sec for duration

In [16]:

```python
import time
import datetime
```

In [17]:

```python
def checki(x):
    y = x[2:]
    h = ''
    m = ''
    s = ''
    nm = ''
    p = ['H', 'M', 'S']
    for i in y:
        if i not in p:
            nm+=i
        else:
            if(i=="H"):
                h = nm
                nm = ''
            elif(i=="H"):
                m = nm
                nm = ''
            else:
                s = nm
                nm = ''
    if(h==''):
        h = '00'
    if(m == ''):
        m = '00'
    if(s==''):
        s = '00'
    bp = h+':'+m+':'+s
    return bp
```

In [20]:

```python
train = pd.read_csv("youtubead_train.csv")
mp = pd.read_csv(path + "youtubead_train.csv")["duration"]
time = mp.apply(checki)
```

In [23]:

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

In [24]:

```python
time1 = time.apply(func_sec)
```

In [25]:

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

Out[25]:

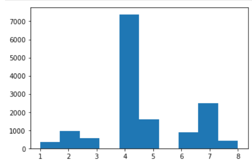
	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	5912	40	1031602	8523	363	1095	2235	37	6
1	2741	2	1707	56	2	6	207	30	4
2	8138	1	2023	25	0	2	1905	16	3
3	9005	6	620860	777	161	153	1952	22	8
4	122	1	666	1	0	0	1783	31	4

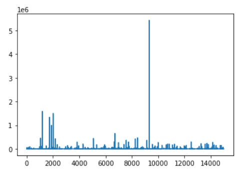
VISUALISATION

Individual plots

In [26]:

```python
plt.hist(data_train["category"])
plt.show()
plt.plot(data_train["adview"])
plt.show()
```





Remove videos with adview greater than 2000000 as outlier

In [27]:

```python
data_train = data_train[data_train["adview"]<2000000]
```

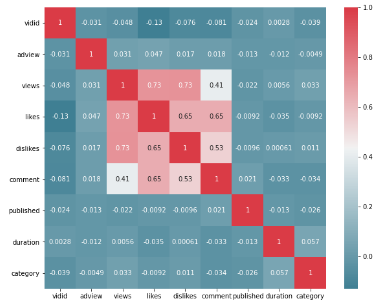
In [28]:

```python
# Heatmap
import seaborn as sns
```

In [30]:

```python
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()
```

```
<ipython-input-30-c41adfee4e44>:3: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing t
his will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  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)
```



Split data

In [31]:

```python
Y_train = pd.DataFrame(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()
```

Out[31]:

	views	likes	dislikes	comment	published	duration	category
0	1031602	8523	363	1095	2235	37	6
1	1707	56	2	6	207	30	4
2	2023	25	0	2	1905	16	3
3	620860	777	161	153	1952	22	8
4	666	1	0	0	1783	31	4

In [32]:

```python
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)
```

In [33]:

```python
X_train.shape
```

Out[33]:

```
(11708, 7)
```

Normalise Data

In [34]:

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

Evaluation Metrics

In [49]:

```
from sklearn import metrics
def print_error(X_test, y_test, model_name):
    prediction = model_name.predict(X_test)
    print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, prediction))
    print('Mean Squared Error:', metrics.mean_squared_error(y_test, prediction))
    print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

Linear Regression

In [50]:

```
from sklearn import linear_model
linear_regression = linear_model.LinearRegression()
linear_regression.fit(X_train, y_train)
print_error(X_test, y_test, linear_regression)
```

```
Mean Absolute Error: 3486.914249706325
Mean Squared Error: 838702210.0737183
Root Mean Squared Error: 28960.35583472203
```

Decision Tree Regressor

In [51]:

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

```
Mean Absolute Error: 4523.868169398907
Mean Squared Error: 2808747537.045765
Root Mean Squared Error: 52997.618220498975
```

Random Forest Regressor

In [52]:

```
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_split, min_samples_leaf= min_samples_leaf)
random_forest.fit(X_train, y_train)
print_error(X_test, y_test, random_forest)
```

```
<ipython-input-52-afdaf1848bcb>:7: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), fo
r example using ravel().
  random_forest.fit(X_train, y_train)
Mean Absolute Error: 3641.7688620591857
Mean Squared Error: 755105855.4409146
Root Mean Squared Error: 27479.18949752548
```

Support Vector Regressor

In [53]:

```
from sklearn.svm import SVR
supportvector_regressor = SVR()
supportvector_regressor.fit(X_train,y_train)
print_error(X_test, y_test, linear_regression)
```

```
C:\Users\Srinivas\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
Mean Absolute Error: 3486.914249706325
Mean Squared Error: 838702210.0737183
Root Mean Squared Error: 28960.35583472203
```

Artificial Neural Network

In [ ]:

```
import keras
```

In [ ]:

```

```