Objective

To build a machine learning regression to predict youtube adview count based on other youtube metrics. Features of data frame: 'views' : The number of unique views for each video 'likes' : The number of likes for each video 'dislikes' : The number of likes for each video 'comment' : The number of unique comments for each video 'published' : The data of uploading the video 'duration' : The duration of the video (in min. and seconds) 'category' : Category niche of each of the video

In [32]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Importing data

In [3]:
```
data = pd.read_csv("train.csv")
data.head()
```

Out[3]:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	F
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	D
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	C
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	H
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	D

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   vidid      14999 non-null  object
 1   adview     14999 non-null  int64
 2   views      14999 non-null  object
 3   likes      14999 non-null  object
 4   dislikes   14999 non-null  object
 5   comment    14999 non-null  object
 6   published  14999 non-null  object
 7   duration   14999 non-null  object
 8   category   14999 non-null  object
dtypes: int64(1), object(8)
memory usage: 1.0+ MB
```

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
vidid        0
adview       0
views        0
likes        0
dislikes     0
comment      0
published    0
duration     0
category     0
dtype: int64
```

In [6]:
```
data.shape
```

Out[6]:
```
(14999, 9)
```

Assigning each category a number for Category feature

In [7]:
```
category = {'A':1,'B':2 ,'C':3,'D':4,'E':5,'F':6,'G':7,'H':8}
data["category"]=data["category"].map(category)
data.head()
```

Out[7]:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	6
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	4
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	3
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	8
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	4

removing category "F" present in data

In [8]:
```
data =data[data.views!='F']
data =data[data.likes!='F']
data =data[data.dislikes!='F']
data =data[data.comment!='F']
```

In [9]:
```
data.head()
```

Out[9]:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	6
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	4
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	3
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	8
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	4

convert values to integer for view ,likes ,comment ,dislikes adnd adview

In [10]:
```
data["views"]=pd.to_numeric(data["views"])
data["comment"]=pd.to_numeric(data["comment"])
data["dislikes"]=pd.to_numeric(data["dislikes"])
data["likes"]=pd.to_numeric(data["likes"])
data["adview"]=pd.to_numeric(data["adview"])
```

In [11]:
```
column_vidid= data['vidid']
```

Encoding features like Category , Duration, Vidid

In [12]:
```
from sklearn.preprocessing import LabelEncoder
data['duration']= LabelEncoder().fit_transform(data['duration'])
data['vidid']= LabelEncoder().fit_transform(data['vidid'])
data['published']= LabelEncoder().fit_transform(data['published'])
```

In [13]:
```
data.head()
```

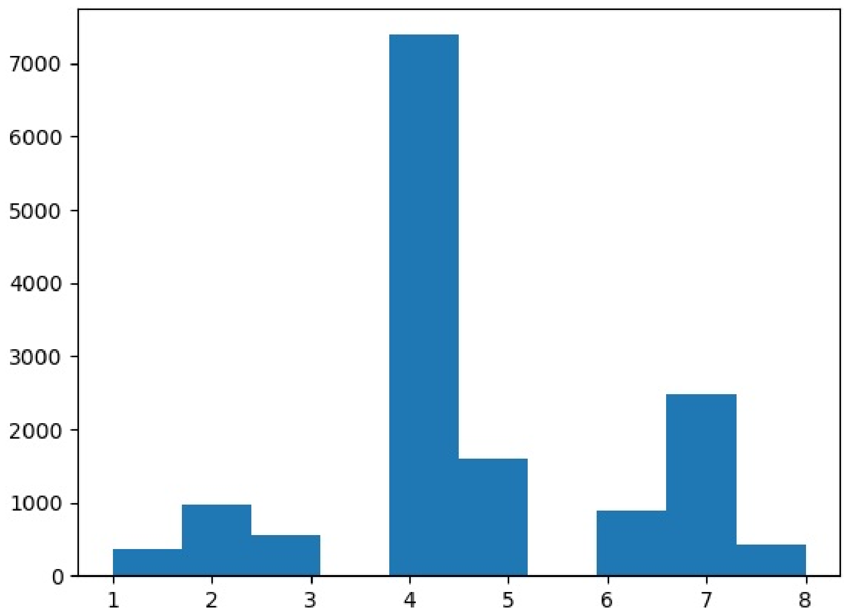
Out[13]:

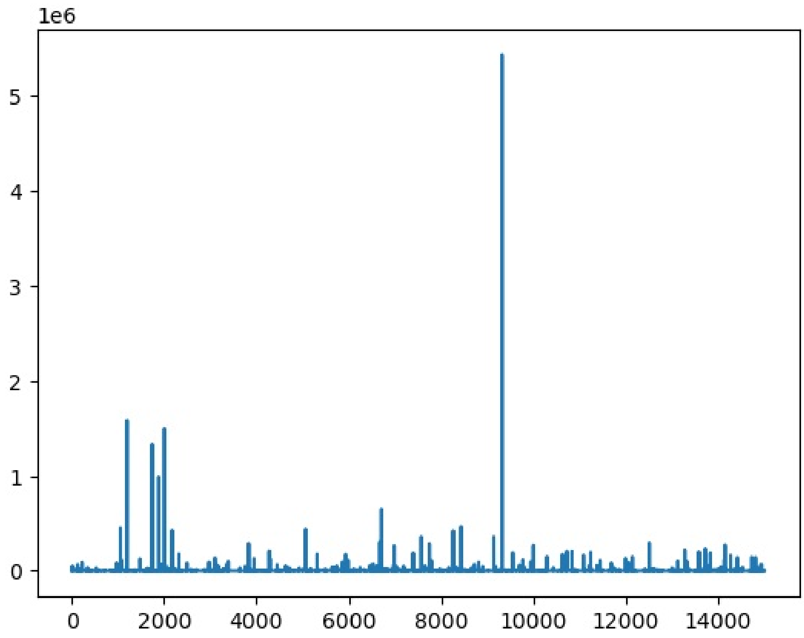
	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	5912	40	1031602	8523	363	1095	2168	2925	6
1	2741	2	1707	56	2	6	2185	3040	4
2	8138	1	2023	25	0	2	2094	1863	3
3	9005	6	620860	777	161	153	2119	2546	8
4	122	1	666	1	0	0	2091	1963	4

visualisation

In [14]:
```
# induvidual plots

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





In [ ]:

```python
# removing vedioes with adview greater than 2000000  as outlier

data = data[data["adview"] < 2000000]
```

In [16]:

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

In [17]:

```python
data.corr()
```

Out[17]:

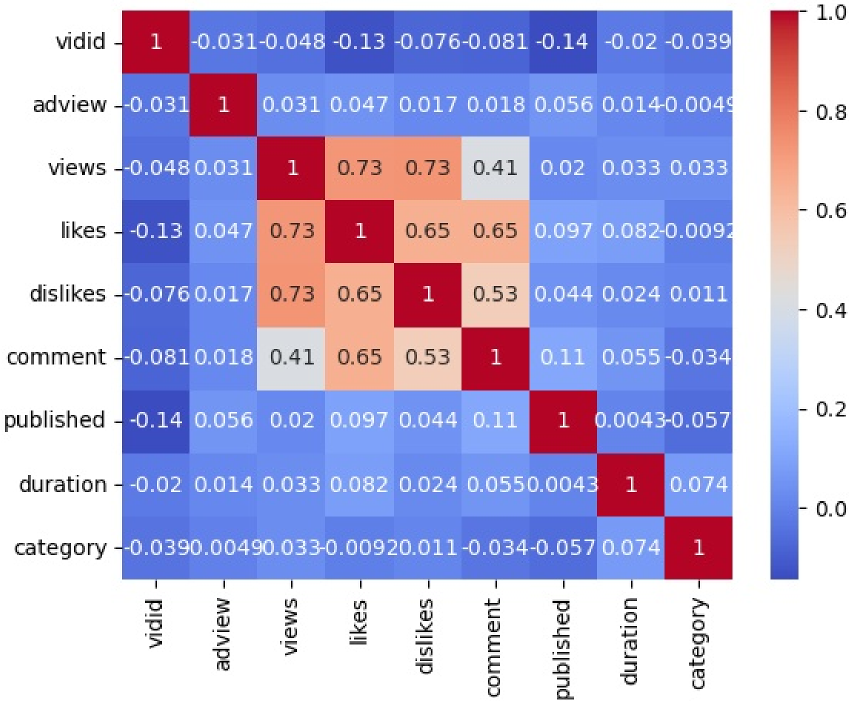
	vidid	adview	views	likes	dislikes	comment	published	duration	category
vidid	1.000000	-0.031080	-0.047581	-0.128860	-0.076460	-0.081059	-0.144470	-0.019815	-0.038917
adview	-0.031080	1.000000	0.031177	0.046541	0.016686	0.017631	0.055657	0.014143	-0.004910
views	-0.047581	0.031177	1.000000	0.726599	0.730216	0.410597	0.020110	0.033191	0.032822
likes	-0.128860	0.046541	0.726599	1.000000	0.648631	0.651215	0.096941	0.081991	-0.009175
dislikes	-0.076460	0.016686	0.730216	0.648631	1.000000	0.532588	0.043745	0.023553	0.011355
comment	-0.081059	0.017631	0.410597	0.651215	0.532588	1.000000	0.114253	0.055479	-0.034107
published	-0.144470	0.055657	0.020110	0.096941	0.043745	0.114253	1.000000	0.004347	-0.056814
duration	-0.019815	0.014143	0.033191	0.081991	0.023553	0.055479	0.004347	1.000000	0.073838
category	-0.038917	-0.004910	0.032822	-0.009175	0.011355	-0.034107	-0.056814	0.073838	1.000000

In [18]:

```python
sns.heatmap(data.corr(),annot = True ,cmap = 'coolwarm')
```

Out[18]:

<AxesSubplot:>



Split data

In [19]:
```
Y_train = pd.DataFrame(data = data.iloc[:, 1].values,columns = ['target'])
data = data.drop(["adview"],axis=1)
data = data.drop(["vidid"],axis=1)
data.head()
```

Out[19]:

	views	likes	dislikes	comment	published	duration	category
0	1031602	8523	363	1095	2168	2925	6
1	1707	56	2	6	2185	3040	4
2	2023	25	0	2	2094	1863	3
3	620860	777	161	153	2119	2546	8
4	666	1	0	0	2091	1963	4

In [20]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(data , Y_train , test_size=0.2 , random_state = 42)
```

Normalise Data

In [21]:
```
X_train.shape
```

Out[21]:
```
(11708, 7)
```

In [22]:
```
Y_train.shape
```

Out[22]:
```
(11708, 1)
```

In [23]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train - scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [24]:
```
X_train.mean()
```

Out[24]:
```
views        697416.108558
likes          2771.122139
dislikes        251.030577
comment         418.178767
published      1551.810215
duration       1848.381363
category          4.611548
dtype: float64
```

Evaluation Metrics

In [25]:
```
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 [26]:
```
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: 4810.954074031322
Mean Squared error 853969395.112762
Root Mean Squared error 29222.7547488727
```

Decision Tree regression

In [27]:
```
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: 3701.9853142076504
Mean Squared error 831060383.9436475
Root Mean Squared error 28828.117939672153
```

Support Vector Regressor

In [ ]:
```
from sklearn.svm import SVR
supportvector_regressor = SVR()
supportvector_regressor.fit(X_train,Y_train)
print_error(X_test,Y_test,linear_regression)
```

Artificial Neural Network

In [29]:
```
import keras
from keras.layers import Dense

ann = keras.models.Sequential([
    Dense(6, activation = "relu",
         input_shape=X_train.shape[1:]),
    Dense(6,activation="relu"),
    Dense(1)
])

optimizer=keras.optimizers.Adam()
loss=keras.losses.mean_squared_error
ann.compile(optimizer=optimizer,loss=loss,metrics=["mean_squared_error"])

history=ann.fit(X_train,Y_train,epochs=100)

ann.summary()

print_error(X_test,Y_test,ann)
```

```
Epoch 1/100
366/366 [==============================] - 3s 3ms/step - loss: 489369698304.0000 - mean_squared_error: 48936969
8304.0000
```

```
Epoch 2/100
366/366 [==============================] - 1s 3ms/step - loss: 50199109632.0000 - mean_squared_error: 501991096
32.0000
Epoch 3/100
366/366 [==============================] - 1s 3ms/step - loss: 1354062976.0000 - mean_squared_error: 1354062976
.0000
Epoch 4/100
366/366 [==============================] - 1s 3ms/step - loss: 764315008.0000 - mean_squared_error: 764315008.0
000
Epoch 5/100
366/366 [==============================] - 1s 3ms/step - loss: 763703040.0000 - mean_squared_error: 763703040.0
000
Epoch 6/100
366/366 [==============================] - 1s 3ms/step - loss: 764862464.0000 - mean_squared_error: 764862464.0
000
Epoch 7/100
366/366 [==============================] - 1s 3ms/step - loss: 764876608.0000 - mean_squared_error: 764876608.0
000
Epoch 8/100
366/366 [==============================] - 1s 3ms/step - loss: 769001408.0000 - mean_squared_error: 769001408.0
000
Epoch 9/100
366/366 [==============================] - 1s 3ms/step - loss: 767656128.0000 - mean_squared_error: 767656128.0
000
Epoch 10/100
366/366 [==============================] - 1s 4ms/step - loss: 766314880.0000 - mean_squared_error: 766314880.0
000
Epoch 11/100
366/366 [==============================] - 1s 3ms/step - loss: 772016256.0000 - mean_squared_error: 772016256.0
000
Epoch 12/100
366/366 [==============================] - 1s 3ms/step - loss: 775430656.0000 - mean_squared_error: 775430656.0
000
Epoch 13/100
366/366 [==============================] - 1s 3ms/step - loss: 769585920.0000 - mean_squared_error: 769585920.0
000
Epoch 14/100
366/366 [==============================] - 1s 3ms/step - loss: 775051456.0000 - mean_squared_error: 775051456.0
000
Epoch 15/100
366/366 [==============================] - 1s 3ms/step - loss: 782586176.0000 - mean_squared_error: 782586176.0
000
Epoch 16/100
366/366 [==============================] - 1s 3ms/step - loss: 803080064.0000 - mean_squared_error: 803080064.0
000
Epoch 17/100
366/366 [==============================] - 1s 3ms/step - loss: 800915136.0000 - mean_squared_error: 800915136.0
000
Epoch 18/100
366/366 [==============================] - 1s 3ms/step - loss: 784023168.0000 - mean_squared_error: 784023168.0
000
Epoch 19/100
366/366 [==============================] - 1s 3ms/step - loss: 932848448.0000 - mean_squared_error: 932848448.0
000
Epoch 20/100
366/366 [==============================] - 1s 3ms/step - loss: 3630356992.0000 - mean_squared_error: 3630356992
.0000
Epoch 21/100
366/366 [==============================] - 1s 3ms/step - loss: 813154496.0000 - mean_squared_error: 813154496.0
000
Epoch 22/100
366/366 [==============================] - 1s 3ms/step - loss: 819214080.0000 - mean_squared_error: 819214080.0
000
Epoch 23/100
366/366 [==============================] - 1s 3ms/step - loss: 786995968.0000 - mean_squared_error: 786995968.0
000
Epoch 24/100
366/366 [==============================] - 1s 2ms/step - loss: 799425984.0000 - mean_squared_error: 799425984.0
000
Epoch 25/100
366/366 [==============================] - 1s 3ms/step - loss: 806112512.0000 - mean_squared_error: 806112512.0
000
Epoch 26/100
366/366 [==============================] - 1s 2ms/step - loss: 788375616.0000 - mean_squared_error: 788375616.0
000
Epoch 27/100
366/366 [==============================] - 1s 3ms/step - loss: 815725952.0000 - mean_squared_error: 815725952.0
000
Epoch 28/100
366/366 [==============================] - 1s 3ms/step - loss: 801998464.0000 - mean_squared_error: 801998464.0
000
Epoch 29/100
366/366 [==============================] - 1s 3ms/step - loss: 886251456.0000 - mean_squared_error: 886251456.0
000
Epoch 30/100
366/366 [==============================] - 1s 4ms/step - loss: 829438208.0000 - mean_squared_error: 829438208.0
000
Epoch 31/100
366/366 [==============================] - 1s 3ms/step - loss: 1057680704.0000 - mean_squared_error: 1057680704
```

```
.0000
Epoch 32/100
366/366 [==============================] - 1s 3ms/step - loss: 909932096.0000 - mean_squared_error: 909932096.0
000
Epoch 33/100
366/366 [==============================] - 1s 3ms/step - loss: 1573473152.0000 - mean_squared_error: 1573473152
.0000
Epoch 34/100
366/366 [==============================] - 1s 3ms/step - loss: 848002176.0000 - mean_squared_error: 848002176.0
000
Epoch 35/100
366/366 [==============================] - 1s 3ms/step - loss: 788164672.0000 - mean_squared_error: 788164672.0
000
Epoch 36/100
366/366 [==============================] - 1s 3ms/step - loss: 848529664.0000 - mean_squared_error: 848529664.0
000
Epoch 37/100
366/366 [==============================] - 1s 3ms/step - loss: 869426432.0000 - mean_squared_error: 869426432.0
000
Epoch 38/100
366/366 [==============================] - 1s 3ms/step - loss: 806428544.0000 - mean_squared_error: 806428544.0
000
Epoch 39/100
366/366 [==============================] - 1s 3ms/step - loss: 2244427520.0000 - mean_squared_error: 2244427520
.0000
Epoch 40/100
366/366 [==============================] - 1s 3ms/step - loss: 801430720.0000 - mean_squared_error: 801430720.0
000
Epoch 41/100
366/366 [==============================] - 1s 3ms/step - loss: 860939392.0000 - mean_squared_error: 860939392.0
000
Epoch 42/100
366/366 [==============================] - 1s 3ms/step - loss: 838277376.0000 - mean_squared_error: 838277376.0
000
Epoch 43/100
366/366 [==============================] - 1s 3ms/step - loss: 863363264.0000 - mean_squared_error: 863363264.0
000
Epoch 44/100
366/366 [==============================] - 2s 5ms/step - loss: 936751552.0000 - mean_squared_error: 936751552.0
000
Epoch 45/100
366/366 [==============================] - 1s 4ms/step - loss: 1018278464.0000 - mean_squared_error: 1018278464
.0000
Epoch 46/100
366/366 [==============================] - 1s 4ms/step - loss: 866351360.0000 - mean_squared_error: 866351360.0
000
Epoch 47/100
366/366 [==============================] - 1s 3ms/step - loss: 865088576.0000 - mean_squared_error: 865088576.0
000
Epoch 48/100
366/366 [==============================] - 1s 3ms/step - loss: 842088768.0000 - mean_squared_error: 842088768.0
000
Epoch 49/100
366/366 [==============================] - 1s 4ms/step - loss: 1283190912.0000 - mean_squared_error: 1283190912
.0000
Epoch 50/100
366/366 [==============================] - 1s 3ms/step - loss: 930555968.0000 - mean_squared_error: 930555968.0
000
Epoch 51/100
366/366 [==============================] - 1s 3ms/step - loss: 909385344.0000 - mean_squared_error: 909385344.0
000
Epoch 52/100
366/366 [==============================] - 1s 3ms/step - loss: 835366400.0000 - mean_squared_error: 835366400.0
000
Epoch 53/100
366/366 [==============================] - 1s 3ms/step - loss: 967487872.0000 - mean_squared_error: 967487872.0
000
Epoch 54/100
366/366 [==============================] - 1s 3ms/step - loss: 1955065600.0000 - mean_squared_error: 1955065600
.0000
Epoch 55/100
366/366 [==============================] - 1s 3ms/step - loss: 792958656.0000 - mean_squared_error: 792958656.0
000
Epoch 56/100
366/366 [==============================] - 1s 3ms/step - loss: 4103665152.0000 - mean_squared_error: 4103665152
.0000
Epoch 57/100
366/366 [==============================] - 1s 3ms/step - loss: 763200512.0000 - mean_squared_error: 763200512.0
000
Epoch 58/100
366/366 [==============================] - 1s 3ms/step - loss: 800743552.0000 - mean_squared_error: 800743552.0
000
Epoch 59/100
366/366 [==============================] - 1s 3ms/step - loss: 787137280.0000 - mean_squared_error: 787137280.0
000
Epoch 60/100
366/366 [==============================] - 1s 3ms/step - loss: 782145472.0000 - mean_squared_error: 782145472.0
000
Epoch 61/100
```

```
366/366 [==============================] - 1s 3ms/step - loss: 831674880.0000 - mean_squared_error: 831674880.0
000
Epoch 62/100
366/366 [==============================] - 1s 3ms/step - loss: 1048356416.0000 - mean_squared_error: 1048356416
.0000
Epoch 63/100
366/366 [==============================] - 1s 3ms/step - loss: 1970857728.0000 - mean_squared_error: 1970857728
.0000
Epoch 64/100
366/366 [==============================] - 1s 3ms/step - loss: 1675658496.0000 - mean_squared_error: 1675658496
.0000
Epoch 65/100
366/366 [==============================] - 1s 3ms/step - loss: 784333568.0000 - mean_squared_error: 784333568.0
000
Epoch 66/100
366/366 [==============================] - 1s 3ms/step - loss: 779276928.0000 - mean_squared_error: 779276928.0
000
Epoch 67/100
366/366 [==============================] - 1s 3ms/step - loss: 1126430592.0000 - mean_squared_error: 1126430592
.0000
Epoch 68/100
366/366 [==============================] - 1s 3ms/step - loss: 801468864.0000 - mean_squared_error: 801468864.0
000
Epoch 69/100
366/366 [==============================] - 1s 3ms/step - loss: 838972672.0000 - mean_squared_error: 838972672.0
000
Epoch 70/100
366/366 [==============================] - 1s 3ms/step - loss: 819810176.0000 - mean_squared_error: 819810176.0
000
Epoch 71/100
366/366 [==============================] - 1s 3ms/step - loss: 799933248.0000 - mean_squared_error: 799933248.0
000
Epoch 72/100
366/366 [==============================] - 1s 3ms/step - loss: 1070606208.0000 - mean_squared_error: 1070606208
.0000
Epoch 73/100
366/366 [==============================] - 1s 3ms/step - loss: 929272000.0000 - mean_squared_error: 929272000.0
000
Epoch 74/100
366/366 [==============================] - 1s 3ms/step - loss: 812118720.0000 - mean_squared_error: 812118720.0
000
Epoch 75/100
366/366 [==============================] - 1s 3ms/step - loss: 838704960.0000 - mean_squared_error: 838704960.0
000
Epoch 76/100
366/366 [==============================] - 1s 3ms/step - loss: 3029281280.0000 - mean_squared_error: 3029281280
.0000
Epoch 77/100
366/366 [==============================] - 1s 3ms/step - loss: 809785280.0000 - mean_squared_error: 809785280.0
000
Epoch 78/100
366/366 [==============================] - 1s 3ms/step - loss: 798712128.0000 - mean_squared_error: 798712128.0
000
Epoch 79/100
366/366 [==============================] - 1s 3ms/step - loss: 857175296.0000 - mean_squared_error: 857175296.0
000
Epoch 80/100
366/366 [==============================] - 1s 3ms/step - loss: 783898624.0000 - mean_squared_error: 783898624.0
000
Epoch 81/100
366/366 [==============================] - 1s 3ms/step - loss: 787750400.0000 - mean_squared_error: 787750400.0
000
Epoch 82/100
366/366 [==============================] - 1s 3ms/step - loss: 1171756032.0000 - mean_squared_error: 1171756032
.0000
Epoch 83/100
366/366 [==============================] - 1s 3ms/step - loss: 826285760.0000 - mean_squared_error: 826285760.0
000
Epoch 84/100
366/366 [==============================] - 1s 3ms/step - loss: 789239296.0000 - mean_squared_error: 789239296.0
000
Epoch 85/100
366/366 [==============================] - 1s 3ms/step - loss: 908592064.0000 - mean_squared_error: 908592064.0
000
Epoch 86/100
366/366 [==============================] - 1s 3ms/step - loss: 797325632.0000 - mean_squared_error: 797325632.0
000
Epoch 87/100
366/366 [==============================] - 1s 3ms/step - loss: 794568512.0000 - mean_squared_error: 794568512.0
000
Epoch 88/100
366/366 [==============================] - 1s 3ms/step - loss: 805236800.0000 - mean_squared_error: 805236800.0
000
Epoch 89/100
366/366 [==============================] - 1s 3ms/step - loss: 840021952.0000 - mean_squared_error: 840021952.0
000
Epoch 90/100
366/366 [==============================] - 1s 3ms/step - loss: 853135488.0000 - mean_squared_error: 853135488.0
000
```

```
Epoch 91/100
366/366 [==============================] - 1s 3ms/step - loss: 1008380352.0000 - mean_squared_error: 1008380352
.0000
Epoch 92/100
366/366 [==============================] - 1s 3ms/step - loss: 826322944.0000 - mean_squared_error: 826322944.0
000
Epoch 93/100
366/366 [==============================] - 1s 3ms/step - loss: 908758208.0000 - mean_squared_error: 908758208.0
000
Epoch 94/100
366/366 [==============================] - 1s 3ms/step - loss: 800446720.0000 - mean_squared_error: 800446720.0
000
Epoch 95/100
366/366 [==============================] - 1s 3ms/step - loss: 1690211200.0000 - mean_squared_error: 1690211200
.0000
Epoch 96/100
366/366 [==============================] - 1s 3ms/step - loss: 776275904.0000 - mean_squared_error: 776275904.0
000
Epoch 97/100
366/366 [==============================] - 1s 3ms/step - loss: 797496128.0000 - mean_squared_error: 797496128.0
000
Epoch 98/100
366/366 [==============================] - 1s 3ms/step - loss: 804852032.0000 - mean_squared_error: 804852032.0
000
Epoch 99/100
366/366 [==============================] - 1s 3ms/step - loss: 780918656.0000 - mean_squared_error: 780918656.0
000
Epoch 100/100
366/366 [==============================] - 1s 3ms/step - loss: 792814080.0000 - mean_squared_error: 792814080.0
000
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 6)                 48

 dense_1 (Dense)             (None, 6)                 42

 dense_2 (Dense)             (None, 1)                 7

=================================================================
Total params: 97 (388.00 Byte)
Trainable params: 97 (388.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
92/92 [==============================] - 0s 2ms/step
Mean Absolute error: 1697.212458999858
Mean Squared error 833689429.4312754
Root Mean Squared error 28873.680566067003
```

Saving Scikitlearn models

In [30]:
```
import joblib
joblib.dump(Decision_tree,"decisiontree_YOUtubeadview.pkl")
```

Out[30]:
```
['decisiontree_YOUtubeadview.pkl']
```

Saving Keras Artificial Neural Network mode

In [31]:
```
ann.save("ann_youtubeadview.h5")
```

In [ ]:
```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js