In [41]:
```
import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import warnings
```

In [43]:
```
# Importing data
data_train = pd.read_csv("train.csv",encoding='utf-8')
```

In [44]:
```
data_train.head()
```

Out[44]:

	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 [45]:
```
data_train.shape
#(14999,9)
```

Out[45]:
```
(14999, 9)
```

In [46]:
```
# 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()
```

Out[46]:

	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

In [47]:
```
# 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']
```

In [48]:
```
# Convert value to intergers for view, likes , comments, dislikes, and advi
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']
```

In [49]:
```
# Encoding features like Category, Duration
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()
```

Out[49]:

	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

In [50]:
```
# 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("train.csv")["duration"]
time = mp.apply(checki)
```

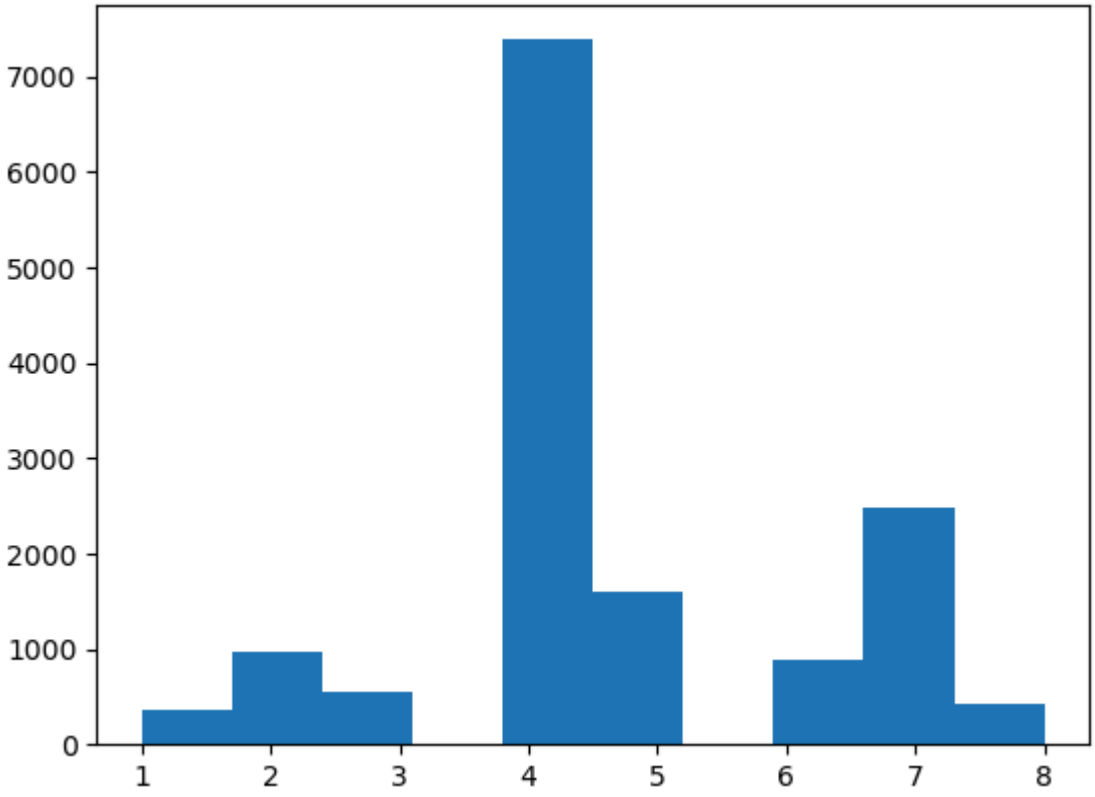
In [51]:

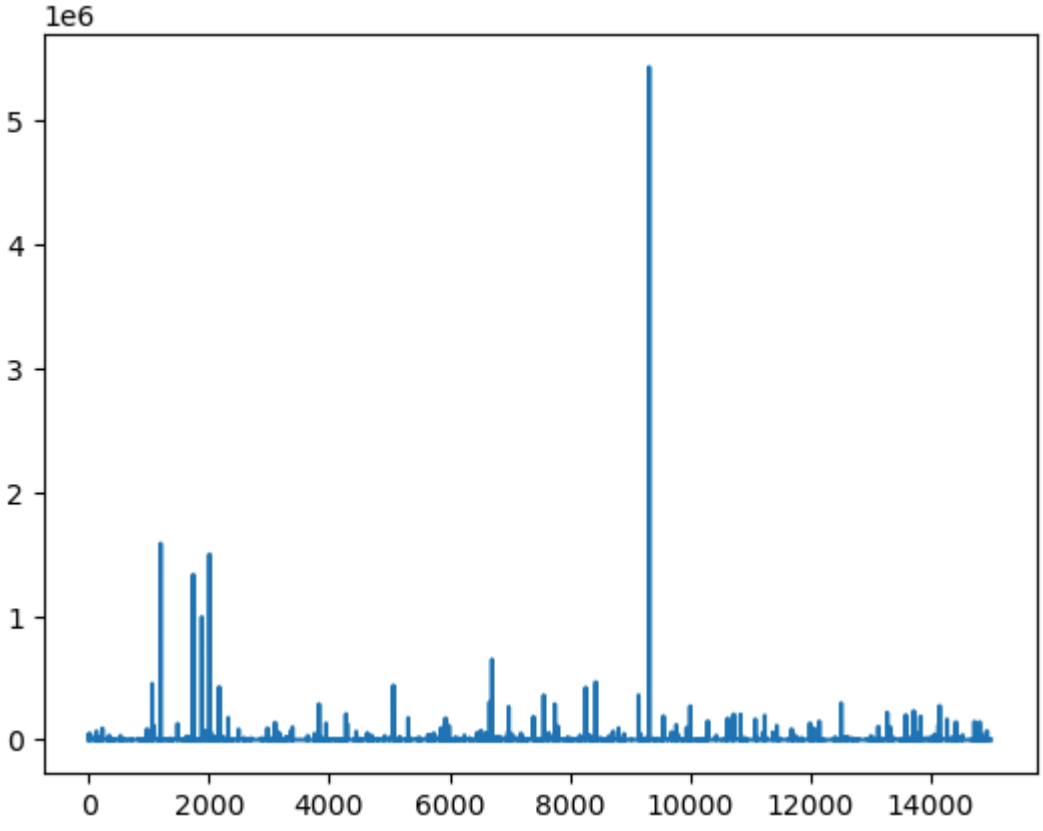
```
train=pd.read_csv("train.csv")
mp=pd.read_csv("train.csv")["duration"]
time=mp.apply(checki)

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(5)
```

Out[51]:

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	5912	40	1031602	8523	363	1095	2168	457	6
1	2741	2	1707	56	2	6	2185	570	4
2	8138	1	2023	25	0	2	2094	136	3
3	9005	6	620860	777	161	153	2119	262	8
4	122	1	666	1	0	0	2091	31	4

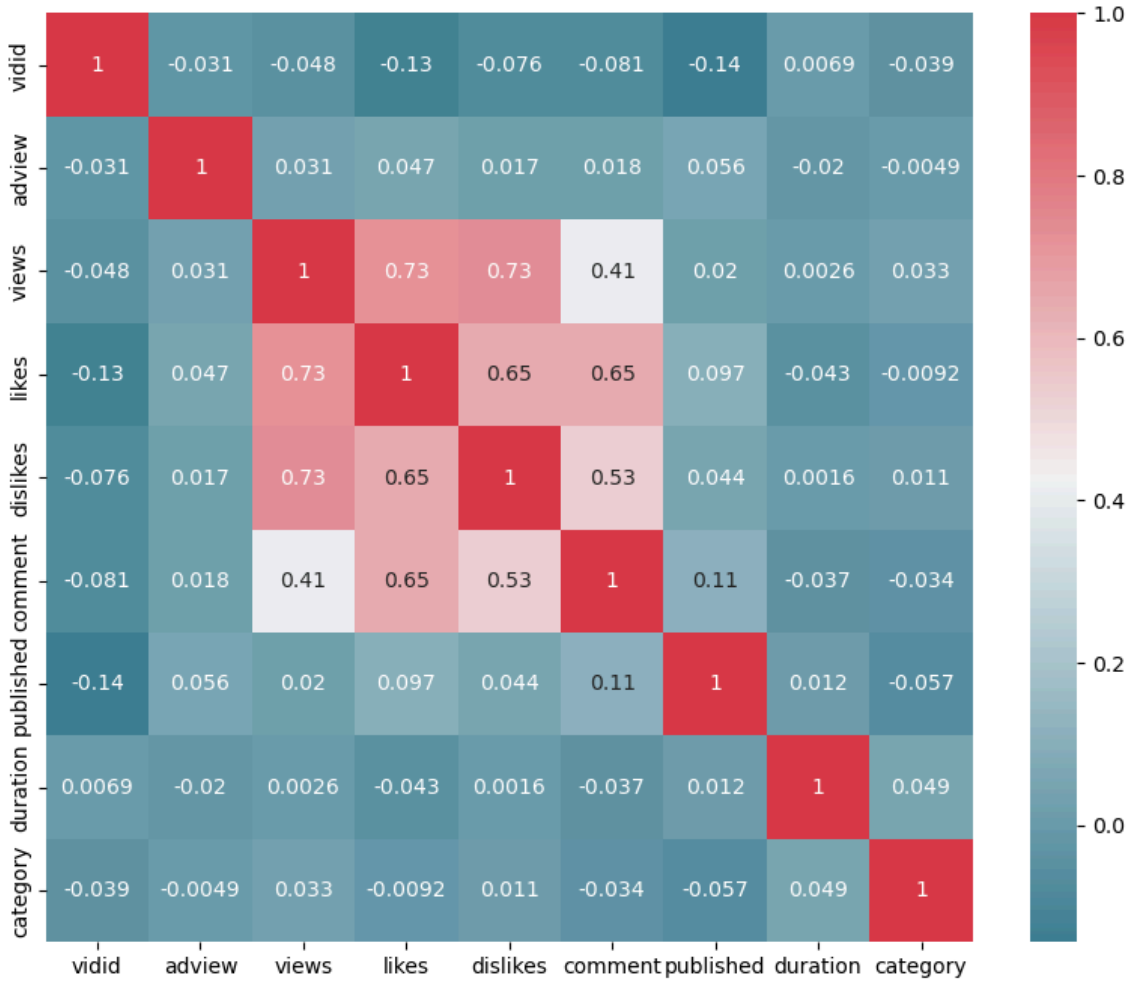
In [52]:
```
#Visualization
   # Individual Plots
plt.hist(data_train["category"])
plt.show()
plt.plot(data_train["adview"])
plt.show()
```





In [53]:
```
#  Remove vedios with adview greater than 2000000 as outlier
data_train = data_train[data_train["adview"]<2000000]
```

In [54]:
```
#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=bool), cmap=sns.diverging_
            square=True, ax=ax, annot=True )
plt.show()
```



In [55]:
```
# Split Data
Y_train = pd.DataFrame(data = data_train.iloc[:, 1].values, columns = ['tar
data_train=data_train.drop(["adview"],axis=1)
data_train=data_train.drop(["vidid"],axis=1)
data_train.head()
```

Out[55]:

	views	likes	dislikes	comment	published	duration	category
0	1031602	8523	363	1095	2168	457	6
1	1707	56	2	6	2185	570	4
2	2023	25	0	2	2094	136	3
3	620860	777	161	153	2119	262	8
4	666	1	0	0	2091	31	4

In [56]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_train, Y_train, te
X_train.shape
```

Out[56]:
```
(11708, 7)
```

In [57]:
```
# Normalize 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()
```

Out[57]:
```
0.1739096800320488
```

In [58]:
```
X_test
```

Out[58]:
```
array([[7.17372469e-03, 8.63080853e-03, 3.31069610e-03, ...,
        1.54621849e-01, 1.32738468e-02, 4.28571429e-01],
       [3.43438476e-06, 1.06225336e-05, 0.00000000e+00, ...,
        8.59243697e-01, 3.68717968e-04, 5.71428571e-01],
       [4.17243632e-04, 7.32954817e-04, 1.13186191e-04, ...,
        5.37394958e-01, 3.31477453e-02, 1.42857143e-01],
       ...,
       [9.69235691e-04, 1.32781670e-04, 3.39558574e-04, ...,
        2.50840336e-01, 1.48593341e-02, 5.71428571e-01],
       [7.67152853e-04, 3.67008535e-03, 5.37634409e-04, ...,
        8.94117647e-01, 1.32001032e-02, 4.28571429e-01],
       [2.63138694e-03, 1.20034629e-03, 8.20599887e-04, ...,
        1.61764706e-01, 2.49990782e-02, 5.71428571e-01]])
```

In [59]:
```
#Evaluation Metrics
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, predi
    print('Mean Squared Error:', metrics.mean_squared_error(y_test, predict
    print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_
```

In [60]:
```
# Linear Regression
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: 3707.378005824534
Mean Squared Error: 835663131.1210337
Root Mean Squared Error: 28907.83857573986
```

In [61]:
```
# 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)
```

```
Mean Absolute Error: 2575.620901639344
Mean Squared Error: 880786041.0382514
Root Mean Squared Error: 29678.039710167035
```

In [62]:
```
# 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_dept
random_forest.fit(X_train,y_train)
print_error(X_test,y_test, random_forest)
```

```
C:\Users\Prachi\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)

Mean Absolute Error: 3359.7327525298447
Mean Squared Error: 708111677.1049092
Root Mean Squared Error: 26610.36784986087
```

In [63]:
```
# Support Vector Regression
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\Prachi\anaconda3\Lib\site-packages\sklearn\utils\validation.py:11
84: DataConversionWarning: A column-vector y was passed when a 1d array wa
s expected. Please change the shape of y to (n_samples, ), for example usi
ng ravel().
  y = column_or_1d(y, warn=True)

Mean Absolute Error: 3707.378005824534
Mean Squared Error: 835663131.1210337
Root Mean Squared Error: 28907.83857573986
```

In [64]:
```
# Artificial Neural Network
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

C:\Users\Prachi\anaconda3\Lib\site-packages\keras\src\layers\core\dens
e.py:85: UserWarning: Do not pass an `input_shape`/`input_dim` argument
to a layer. When using Sequential models, prefer using an `Input(shape)
` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

366/366 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - loss: 676703360.0000 - mean_
squared_error: 676704832.0000
Epoch 2/100
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 352565344.0000 - mean_
squared_error: 352566752.0000
Epoch 3/100
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 899430528.0000 - mean_
squared_error: 899431936.0000
Epoch 4/100
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 463487456.0000 - mean_
squared_error: 463488896.0000
```

In [68]:
```
# Saving Sckitlearn models
import joblib
joblib.dump(decision_tree, "decisiontree_youtubeadview.pkl")

# Saving Keras Artificial Neural Network model
ann.save('ann_youtubeadview.keras')
```

Testing

In [89]:
```
data_test = pd.read_csv("test.csv",encoding='utf-8')
```

In [90]:
```
data_test.head()
```

Out[90]:

	vidid	views	likes	dislikes	comment	published	duration	category
0	VID_1054	440238	6153	218	1377	2017-02-18	PT7M29S	B
1	VID_18629	1040132	8171	340	1047	2016-06-28	PT6M29S	F
2	VID_13967	28534	31	11	1	2014-03-10	PT37M54S	D
3	VID_19442	1316715	2284	250	274	2010-06-05	PT9M55S	G
4	VID_770	1893173	2519	225	116	2016-09-03	PT3M8S	B

In [91]:
```
from keras.models import load_model
model = load_model('ann_youtubeadview.keras')
```

In [92]:
```
# Removing character "F" present in data
data_test=data_test[data_test.views!='F']
data_test=data_test[data_test.likes!='F']
data_test=data_test[data_test.dislikes!='F']
data_test=data_test[data_test.comment!='F']
```

In [93]:
```
data_test.head()
```

Out[93]:

	vidid	views	likes	dislikes	comment	published	duration	category
0	VID_1054	440238	6153	218	1377	2017-02-18	PT7M29S	B
1	VID_18629	1040132	8171	340	1047	2016-06-28	PT6M29S	F
2	VID_13967	28534	31	11	1	2014-03-10	PT37M54S	D
3	VID_19442	1316715	2284	250	274	2010-06-05	PT9M55S	G
4	VID_770	1893173	2519	225	116	2016-09-03	PT3M8S	B

In [94]:
```
# 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_test["category"]=data_test["category"].map(category)
data_test.head()
```

Out[94]:

	vidid	views	likes	dislikes	comment	published	duration	category
0	VID_1054	440238	6153	218	1377	2017-02-18	PT7M29S	2
1	VID_18629	1040132	8171	340	1047	2016-06-28	PT6M29S	6
2	VID_13967	28534	31	11	1	2014-03-10	PT37M54S	4
3	VID_19442	1316715	2284	250	274	2010-06-05	PT9M55S	7
4	VID_770	1893173	2519	225	116	2016-09-03	PT3M8S	2

In [97]:
```
# Convert value to intergers for view, likes , comments, dislikes, and advi
data_test["views"]=pd.to_numeric(data_test["views"])
data_test["comment"]=pd.to_numeric(data_test["comment"])
data_test["likes"]=pd.to_numeric(data_test["likes"])
data_test["dislikes"]=pd.to_numeric(data_test["dislikes"])

column_vidid=data_test['vidid']
```

In [98]:
```
# Encoding features like Category, Duration
from sklearn.preprocessing import LabelEncoder
data_test['duration']=LabelEncoder().fit_transform(data_test['duration'])
data_test['vidid']=LabelEncoder().fit_transform(data_test['vidid'])
data_test['published']=LabelEncoder().fit_transform(data_test['published'])
data_test.head()
```

Out[98]:

	vidid	views	likes	dislikes	comment	published	duration	category
0	231	1031602.0	6153	218	1377	2053	2115	2
1	3444	1707.0	8171	340	1047	1825	2055	6
2	1593	2023.0	31	11	1	1009	1506	4
3	3775	620860.0	2284	250	274	116	2265	7
4	7644	666.0	2519	225	116	1892	1625	2

In [101]:
```
# 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

test=pd.read_csv("test.csv")
mp = pd.read_csv("test.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_test["duration"]=time1
data_test.head(5)
```

Out[101]:

	vidid	views	likes	dislikes	comment	published	duration	category
0	231	1031602.0	6153	218	1377	2053	449	2
1	3444	1707.0	8171	340	1047	1825	389	6
2	1593	2023.0	31	11	1	1009	2274	4
3	3775	620860.0	2284	250	274	116	595	7
4	7644	666.0	2519	225	116	1892	188	2

In [102]:
```
data_test=data_test.drop(["vidid"],axis=1)
data_test.head()
```

Out[102]:

	views	likes	dislikes	comment	published	duration	category
0	1031602.0	6153	218	1377	2053	449	2
1	1707.0	8171	340	1047	1825	389	6
2	2023.0	31	11	1	1009	2274	4
3	620860.0	2284	250	274	116	595	7
4	666.0	2519	225	116	1892	188	2

In [103]:
```
# Normalize Data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_test=scaler.fit_transform(X_test)
```

In [104]:
```
prediction = model.predict(X_test)
```

```
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

In [105]:
```
prediction=pd.DataFrame(prediction)
prediction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2928 entries, 0 to 2927
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       2928 non-null   float32
dtypes: float32(1)
memory usage: 11.6 KB
```

In [106]:
```
prediction = prediction.rename(columns={0: "Adview"})
```

In [107]:
```
prediction.head()
```

Out[107]:

	Adview
0	472.432129
1	2394.801270
2	1069.735596
3	53.477478
4	2105.695557

In [108]:
```
prediction.to_csv('predictions.csv')
```

In [ ]:
```

```