```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import matplotlib.cm as am
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#importing dataset
path = '/content/drive/MyDrive/ML project/train.csv'
data_train = pd.read_csv(path)


data_train.head()
```

	vidid	adview	views	likes	dislikes	comment	published	duration	categor
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	

Next steps: Generate code with data_train View recommended plots

```
data_train.shape
```

```
(14999, 9)
```

```
#into numerical values
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()
```

	vidid	adview	views	likes	dislikes	comment	published	duration	categor
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	

Next steps: Generate code with data_train View recommended plots

```
#removing character F in data
data_train=data_train[data_train.views!='F']
data_train=data_train[data_train.comment!='F']
data_train=data_train[data_train.dislikes!='F']
data_train=data_train[data_train.likes!='F']
data_train.head()
```

	vidid	adview	views	likes	dislikes	comment	published	duration	categor
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	

Next steps: Generate code with data_train View recommended plots

```
#converting into numerical data
data_train['views']=pd.to_numeric(data_train['views'])
data_train['likes']=pd.to_numeric(data_train['likes'])
data_train['dislikes']=pd.to_numeric(data_train['dislikes'])
data_train['category']=pd.to_numeric(data_train['category'])
data_train['comment']=pd.to_numeric(data_train['comment'])
column_vidid = data_train['vidid']
data_train.head()
```

	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

Next steps: Generate code with data_train View recommended plots

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data_train["duration"] = le.fit_transform(data_train["duration"])
data_train["vidid"] = le.fit_transform(data_train["vidid"])
data_train["published"] = le.fit_transform(data_train["published"])
data_train.head()
```

	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

Next steps: Generate code with data_train View recommended plots

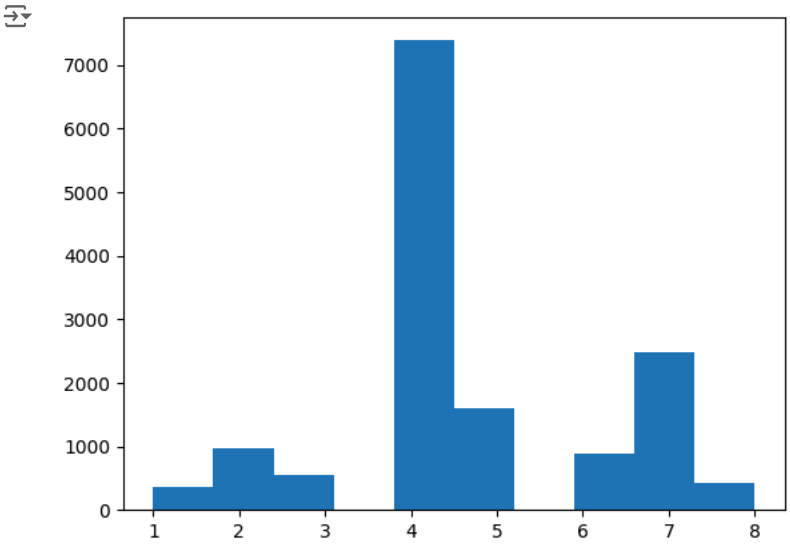
```
# Convert Time_in_sec for duration
import datetime
import time
def checki(x):
  y = x[2:]
  h = ''
  m = ''
  s = ''
  mm = ''
  P = ['H','M','S']
  for i in y:
    if i not in P:
      mm+=i
    else:
      if (i=="H"):
        h = mm
        mm = ''
      elif (i == "M"):
        m = mm
        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(path)
mp = pd.read_csv(path)["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()
```

	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

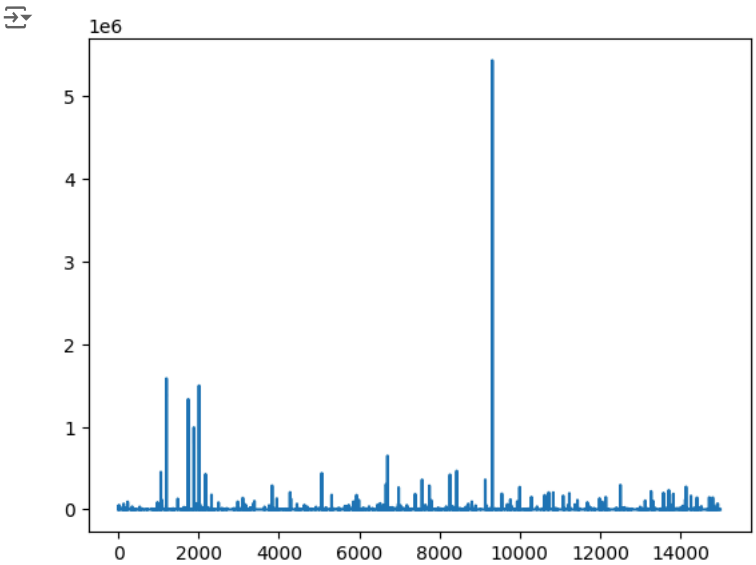
Next steps: Generate code with `data_train` View recommended plots

Visualization

```
plt.hist(data_train['category'])
plt.show()
```

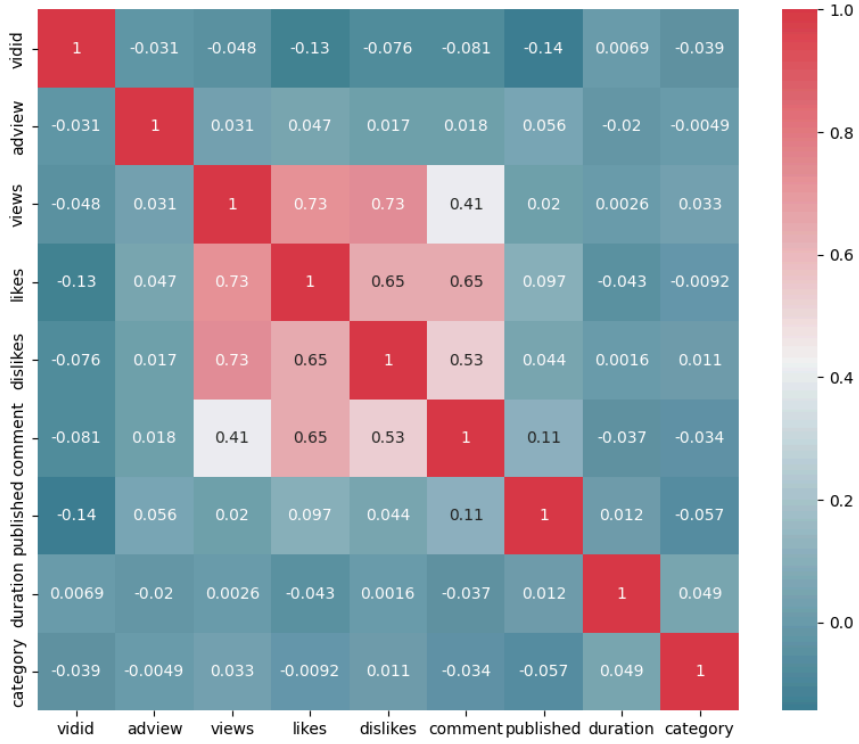



```
plt.plot(data_train['adview'])
plt.show()
```




```
data_train = data_train[data_train['adview']<2000000]
```

```
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_palette(220,10,as_cmap=True),square=True,ax=ax ,annot=Tru
plt.show()
```




```
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()
```

	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

Next steps: Generate code with `data_train` View recommended plots

```
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
```

```
(11708, 7)
```

```
X_test.shape
```

```
(2928, 7)
```

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

X_train.mean()
```

```
0.1739096800320488
```

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

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

```
Mean Absolute Error: 3707.3780058245316
Mean Squared Error: 835663131.1210335
Root Mean Squared Error: 28907.83857573986
```

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

```
Mean Absolute Error: 2653.555669398907
Mean Squared Error: 887036016.9217896
Root Mean Squared Error: 29783.149882471964
```

```
from sklearn.ensemble import RandomForestRegressor
n_estimators = 200
max_depth = 25
min_samples_split = 15
min_samples_leaf = 2
rf = RandomForestRegressor(n_estimators = n_estimators, max_depth = max_depth, min_samples_split = min_samples_split, min_samples_leaf
rf.fit(X_train,y_train)
print_error(X_test,y_test,rf)
```

```
<ipython-input-29-3122f257a706>:7: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change
  rf.fit(X_train,y_train)
Mean Absolute Error: 3505.18676090906
Mean Squared Error: 741977735.859357
Root Mean Squared Error: 27239.268269528773
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when
  y = column_or_1d(y, warn=True)
Mean Absolute Error: 1696.9438599505638
Mean Squared Error: 833685776.029172
Root Mean Squared Error: 28873.617300732723
```

```
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_train,y_train,ann)
```

```
Epoch 85/100
366/366 [==============================] - 1s 2ms/step - loss: 763256576.0000 - mean_squared_error: 763256576.0000
Epoch 86/100
366/366 [==============================] - 1s 2ms/step - loss: 763244096.0000 - mean_squared_error: 763244096.0000
Epoch 87/100
366/366 [==============================] - 1s 2ms/step - loss: 763230976.0000 - mean_squared_error: 763230976.0000
Epoch 88/100
366/366 [==============================] - 1s 2ms/step - loss: 763210240.0000 - mean_squared_error: 763210240.0000
Epoch 89/100
366/366 [==============================] - 1s 2ms/step - loss: 763195648.0000 - mean_squared_error: 763195648.0000
Epoch 90/100
366/366 [==============================] - 1s 2ms/step - loss: 763181952.0000 - mean_squared_error: 763181952.0000
Epoch 91/100
366/366 [==============================] - 1s 2ms/step - loss: 763169984.0000 - mean_squared_error: 763169984.0000
Epoch 92/100
366/366 [==============================] - 1s 2ms/step - loss: 763148928.0000 - mean_squared_error: 763148928.0000
Epoch 93/100
366/366 [==============================] - 1s 2ms/step - loss: 763134976.0000 - mean_squared_error: 763134976.0000
Epoch 94/100
366/366 [==============================] - 1s 2ms/step - loss: 763120256.0000 - mean_squared_error: 763120256.0000
Epoch 95/100
366/366 [==============================] - 1s 2ms/step - loss: 763103808.0000 - mean_squared_error: 763103808.0000
Epoch 96/100
366/366 [==============================] - 1s 2ms/step - loss: 763090048.0000 - mean_squared_error: 763090048.0000
Epoch 97/100
366/366 [==============================] - 1s 2ms/step - loss: 763075136.0000 - mean_squared_error: 763075136.0000
Epoch 98/100
366/366 [==============================] - 1s 2ms/step - loss: 763060544.0000 - mean_squared_error: 763060544.0000
Epoch 99/100
366/366 [==============================] - 1s 2ms/step - loss: 763042368.0000 - mean_squared_error: 763042368.0000
Epoch 100/100
366/366 [==============================] - 1s 2ms/step - loss: 763029440.0000 - mean_squared_error: 763029440.0000
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)
_________________________________________________________________
366/366 [==============================] - 1s 1ms/step
Mean Absolute Error: 3269.3960742431163
Mean Squared Error: 763010750.2725962
Root Mean Squared Error: 27622.64922618025
```

```
score = lr.score(X_test,y_test)
print("Linear Regression: ", score)
score = dt.score(X_test,y_test)
print("decision tree: ",score)
score = rf.score(X_test,y_test)
print("random forest: ",score)
score = svr.score(X_test,y_test)
print("support vector machine: ",score)
```

```
Linear Regression:  -0.005841554563432938
decision tree:  -0.06767625971181412
random forest:  0.10692238116684427
support vector machine:  -0.0034615214550512974
```

```
import joblib
joblib.dump(rf,'rf_Youtube_adview_Prediction.pkl')
```

```
['rf_Youtube_adview_Prediction.pkl']
```

```
ann.save('ann_Youtube_adview_Prediction.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file v
  saving_api.save_model(
```

```
test_data = pd.read_csv('/content/drive/MyDrive/ML project/test.csv')
test_data.head()
```

	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

Next steps: Generate code with test_data View recommended plots

```
#into numerical values
category={'A':1,'B':2,'C':3,'D':4,'E':5,'F':6,'G':7,'H':8}
test_data['category']=test_data['category'].map(category)
test_data.head()
```

	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

Next steps: Generate code with test_data View recommended plots

```
test_data = test_data[test_data.views!='F']
test_data = test_data[test_data.comment!='F']
test_data = test_data[test_data.dislikes!='F']
test_data = test_data[test_data.likes!='F']
test_data.head()
```

	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

Next steps: Generate code with test_data View recommended plots

```
#converting into numerical data
test_data['views']=pd.to_numeric(test_data['views'])
test_data['likes']=pd.to_numeric(test_data['likes'])
test_data['dislikes']=pd.to_numeric(test_data['dislikes'])
test_data['category']=pd.to_numeric(test_data['category'])
test_data['comment']=pd.to_numeric(test_data['comment'])
column_vidid = test_data['vidid']
test_data.head()
```

	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

Next steps: Generate code with test_data View recommended plots