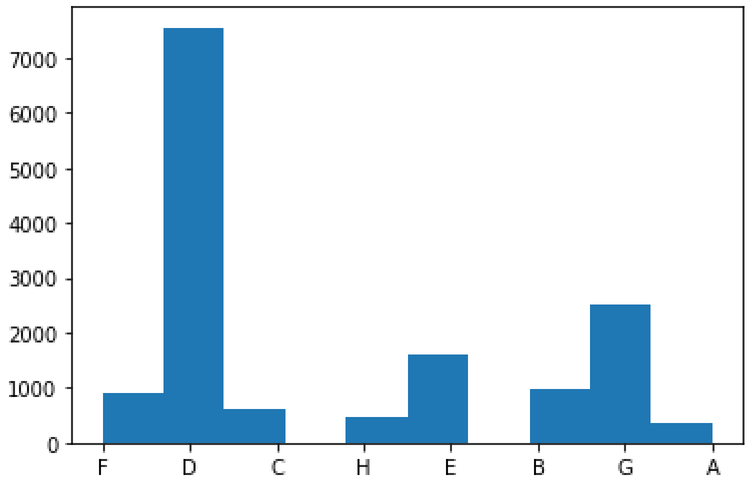
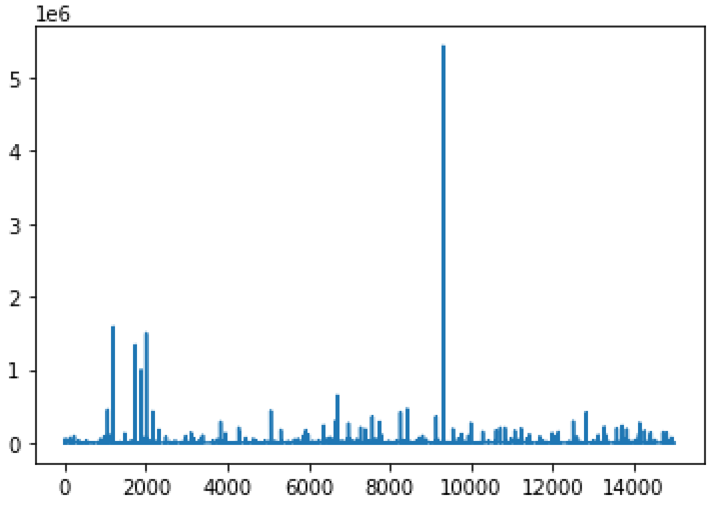
```
import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.pyplot as plt
# Importing data
path = "" # Put path of your folder of your data if it's not in the same folder
data_train = pd.read_csv("train.csv")
data_train.head()
data_train.shape
```

```
(14999, 9)
```

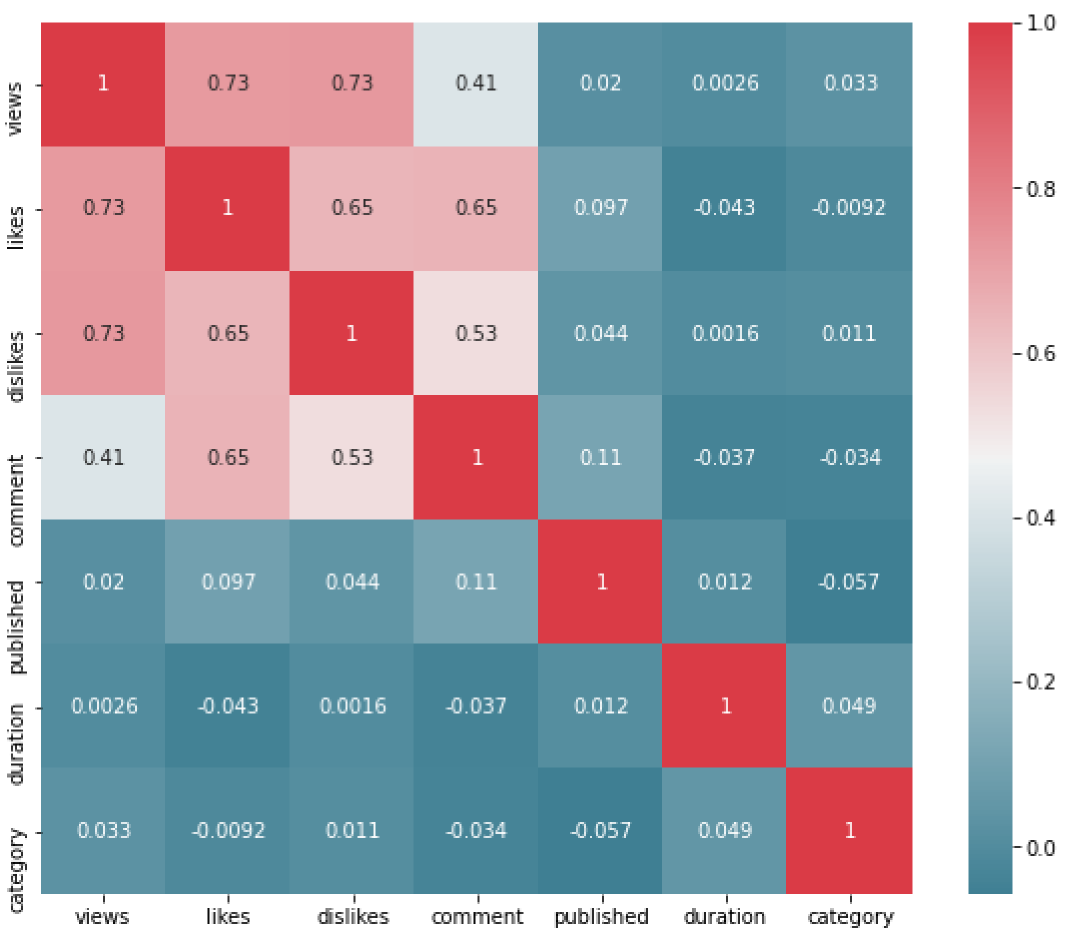
```
# Visualization
# Individual Plots
plt.hist(data_train["category"])
plt.show()
plt.plot(data_train["adview"])
plt.show()
# Remove videos with adview greater than 2000000 as outlier
data_train = data_train[data_train["adview"] <2000000]
```





```
# 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
square=True, ax=ax,annot=True)
plt.show()
```



```
# 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()
```

	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	

```
# 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()
```

```
   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   9004       6   620860    777       161      153       2119      2546         8
4    122       1      666      1         0        0       2091      1963         4
```

```
# 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()
```

	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	9004	6	620860	777	161	153	2119	262	8
4	122	1	666	1	0	0	2091	31	4

```
# Split Data
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

```
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.
X_train.shape
# Normalise 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()
```

```
0.1739096800320488
```

```
# 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, 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, pred
```

```
# 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: 3707.378005824532
Mean Squared Error: 835663131.1210337
Root Mean Squared Error: 28907.83857573986
```

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

```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: DataConversic
  y = column_or_1d(y, warn=True)
Mean Absolute Error: 3707.378005824532
Mean Squared Error: 835663131.1210337
Root Mean Squared Error: 28907.83857573986
```

```
# 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: 2683.9286202185795
Mean Squared Error: 897836055.1950136
Root Mean Squared Error: 29963.91254818058
```

```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
n_estimators = 400
max_depth = 50
min_samples_split= 30
min_samples_leaf= 3
random_forest = RandomForestRegressor(n_estimators = n_estimators, max_depth = max_de
random_forest.fit(X_train,y_train)
print_error(X_test,y_test, random_forest)
print_error(X_test,y_test, random_forest)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DataConversionWarning

Mean Absolute Error: 3416.1310291756986
Mean Squared Error: 742092301.0656787
Root Mean Squared Error: 27241.371130427313
```

```
# 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
366/366 [==============================] - 17s 3ms/step - loss: 469571112.9589 - m
Epoch 2/100
366/366 [==============================] - 1s 3ms/step - loss: 703724705.7231 - me
Epoch 3/100
366/366 [==============================] - 1s 3ms/step - loss: 1227204391.2371 - m
Epoch 4/100
366/366 [==============================] - 1s 3ms/step - loss: 1118232067.2933 - m
Epoch 5/100
366/366 [==============================] - 1s 3ms/step - loss: 661635794.7248 - me
Epoch 6/100
366/366 [==============================] - 1s 3ms/step - loss: 532626680.9373 - me
Epoch 7/100
366/366 [==============================] - 1s 3ms/step - loss: 952438887.0191 - me
Epoch 8/100
366/366 [==============================] - 1s 3ms/step - loss: 1011212140.5443 - m
Epoch 9/100
366/366 [==============================] - 1s 3ms/step - loss: 823545630.3433 - me
Epoch 10/100
366/366 [==============================] - 1s 3ms/step - loss: 794369160.1761 - me
Epoch 11/100
366/366 [==============================] - 1s 3ms/step - loss: 447305384.2603 - me
Epoch 12/100
366/366 [==============================] - 1s 3ms/step - loss: 660571687.4934 - me
Epoch 13/100
366/366 [==============================] - 1s 3ms/step - loss: 476336505.6621 - me
Epoch 14/100
366/366 [==============================] - 1s 3ms/step - loss: 761525071.2071 - me
Epoch 15/100
366/366 [==============================] - 1s 3ms/step - loss: 524057261.6291 - me
Epoch 16/100
366/366 [==============================] - 1s 3ms/step - loss: 1686341799.4114 - m
```

```
Epoch 17/100
366/366 [==============================] - 1s 3ms/step - loss: 608129630.1580 - me
Epoch 18/100
366/366 [==============================] - 1s 3ms/step - loss: 363924695.4537 - me
Epoch 19/100
366/366 [==============================] - 1s 3ms/step - loss: 1045351705.5082 - m
Epoch 20/100
366/366 [==============================] - 1s 3ms/step - loss: 428724058.7420 - me
Epoch 21/100
366/366 [==============================] - 1s 3ms/step - loss: 442341584.7129 - me
Epoch 22/100
366/366 [==============================] - 1s 3ms/step - loss: 407823120.8604 - me
Epoch 23/100
366/366 [==============================] - 1s 3ms/step - loss: 1071861235.8072 - m
Epoch 24/100
366/366 [==============================] - 1s 3ms/step - loss: 581302252.1308 - me
Epoch 25/100
366/366 [==============================] - 1s 3ms/step - loss: 503853939.9108 - me
Epoch 26/100
366/366 [==============================] - 1s 3ms/step - loss: 998389941.9510 - me
Epoch 27/100
366/366 [==============================] - 1s 3ms/step - loss: 743672269.1172 - me
Epoch 28/100
366/366 [==============================] - 1s 3ms/step - loss: 538569881.5228 - me
Epoch 29/100
366/366 [==============================] - 1s 3ms/step - loss: 913507401.8896 - me
```

```
#Saving Scikitlearn models
import joblib
joblib.dump(decision_tree, "decisiontree_youtubeadviewpred.pkl")
# Saving Keras Artificial Neural Network model
ann.save("ann_youtubeadviewpred.h5")
```

```
import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.pyplot as plt
# Importing data
data_test = pd.read_csv("test.csv")
data_test.head()
data_test.shape
```

```
(8764, 8)
```

```
# 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']
data_test.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_test["category"]=data_test["category"].map(category)
data_test.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

```
# Convert values to integers for views, likes, comments, dislikes and adview
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']
```

```
# Endoding features like Category, Duration, Vidid
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()
```

	vidid	views	likes	dislikes	comment	published	duration	category
0	231	440238	6153	218	1377	2053	2115	2
1	3444	1040132	8171	340	1047	1825	2055	6
2	1593	28534	31	11	1	1009	1506	4
3	3775	1316715	2284	250	274	116	2265	7
4	7644	1893173	2519	225	116	1892	1625	2

```
# 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
            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("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()
```

	vidid	views	likes	dislikes	comment	published	duration	category
0	231	440238	6153	218	1377	2053	449	2
1	3444	1040132	8171	340	1047	1825	389	6
2	1593	28534	31	11	1	1009	2274	4
3	3775	1316715	2284	250	274	116	595	7
4	7644	1893173	2519	225	116	1892	188	2

```
# Split Data
Z_train = pd.DataFrame(data = data_test.iloc[:, 1].values, columns = ['target'])
data_test=data_test.drop(["vidid"],axis=1)
data_test.head()
```

	views	likes	dislikes	comment	published	duration	category
0	440238	6153	218	1377	2053	449	2
1	1040132	8171	340	1047	1825	389	6
2	28534	31	11	1	1009	2274	4
3	1316715	2284	250	274	116	595	7
4	1893173	2519	225	116	1892	188	2

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_test, Z_train, test_size=0.2
X_train.shape
# Normalise 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()
```

```
0.1691684916653104
```

```
# 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, 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, pred

```

```
# 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: 3458894.505263158
Mean Squared Error: 115794316199804.33
Root Mean Squared Error: 10760776.747047786
```

```
#Saving Scikitlearn models
import joblib
joblib.dump(decision_tree, "decisiontree_prediction.pkl")
```

```
['decisiontree_prediction.pkl']
```

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

```
y_pred = regressor.predict(X_test)
```

```
df=pd.DataFrame({'Predictions':y_pred})
df.head()
```

	Predictions
0	265021.0
1	14179.0
2	9682759.0
3	24248747.0
4	425196.0

```
predictions=df
```

```
predictions
```

	Predictions
0	265021.0
1	14179.0
2	9682759.0
3	24248747.0
4	425196.0
...	...
1705	13572428.0
1706	199992.0
1707	6011924.0
1708	5662900.0
1709	1005358.0

1710 rows × 1 columns

```
predictions.to_csv('predictions.csv', index=False)
```

```
! cat predictions.csv
```

```
Predictions
265021.0
14179.0
9682759.0
24248747.0
425196.0
356072.0
350410.0
4283780.0
800112.0
28556617.0
3046514.0
6947.0
14957150.0
```

```
27112851.0
29390.0
878075.0
45510.0
543526.0
891984.0
203763.0
430245.0
4006843.0
7841531.0
276.0
259759.0
4401353.0
5767561.0
49727.0
5646790.0
44952.0
1385406.0
3725457.0
292963.0
6376460.0
1333017.0
1621910.0
87814.0
1409323.0
2983.0
1368706.0
356503.0
483380.0
11483.0
1568927.0
216077.0
567505.0
16539356.0
1143033.0
185502.0
1504436.0
327481.0
542673.0
740999.0
1339.0
1511094.0
468635.0
166598.0
1718530.0
```