Welcome to my project

Youtube adview Prediction Project

> Using Train.csv

by - Shreyansh Saagar

User_id = Shreyansh Saagar

```
import numpy as np
import pandas as pd
import matplotlib.cm as cm
import matplotlib.pyplot as plt
```

```
data_train = pd.read_csv( Path + "train.csv" )
data_train.head()
```

	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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2016-07-	[illegible]	[illegible]

```
data_train.shape
```

```
(14999, 9)
```

```
data_train.dtypes
```

```
vidid        object
adview        int64
views        object
likes        object
dislikes     object
comment      object
published    object
duration     object
```

```
    category     object
    dtype: object
```

```
# 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	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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2016-07-	[illegible]	[illegible]

Clean the dataset by removing missing values and other things.

```
# 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()
```

	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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2016-07-	[illegible]	[illegible]

```
data_train.shape
```

```
    (14637, 9)
```

```
data_train.dtypes
```

```
vidid        object
adview        int64
views        object
likes        object
dislikes     object
comment      object
published    object
duration     object
category      int64
dtype: object
```

Transform attributes into numerical values and other necessary transformations

```
# 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	9005	6	620860	777	161	153	2119	2546	8
4	122	1	666	1	0	0	2091	1963	4

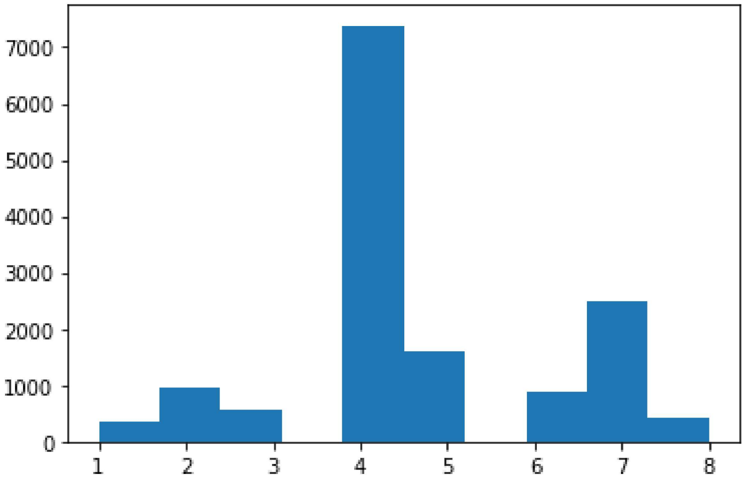
```
# Convert Time_in_sec for duration
import datetime
import time

def checki(x):
```

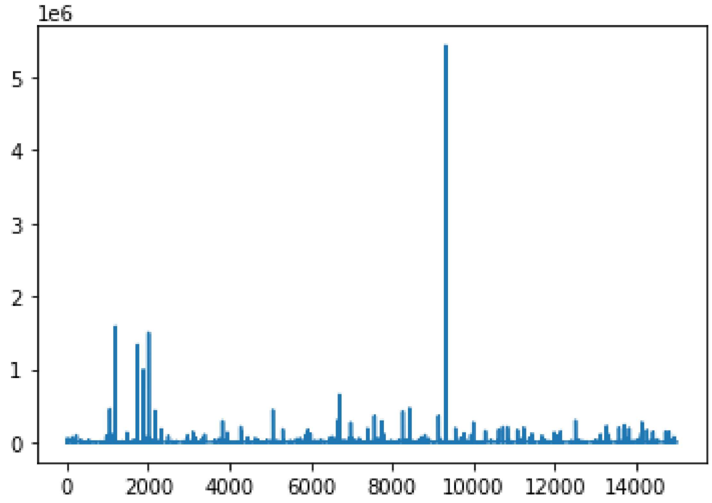
```
def checki(x):
  y = x[2:]
  h = ''
  m = ''
  s = ''
  mm = ''
  P = ['H','M','S']
  for i in y:
    if i not in P:
      mm+=i
    else:
      if(i=="H"):
        h = mm
        mm = ''
      elif(i == "M"):
        m = mm
        mm = ''
      else:
        s = mm
        mm = ''
  if(h==''):
    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   9005       6   620860    777       161      153       2119       262         8
4    122       1      666      1         0        0       2091        31         4
```

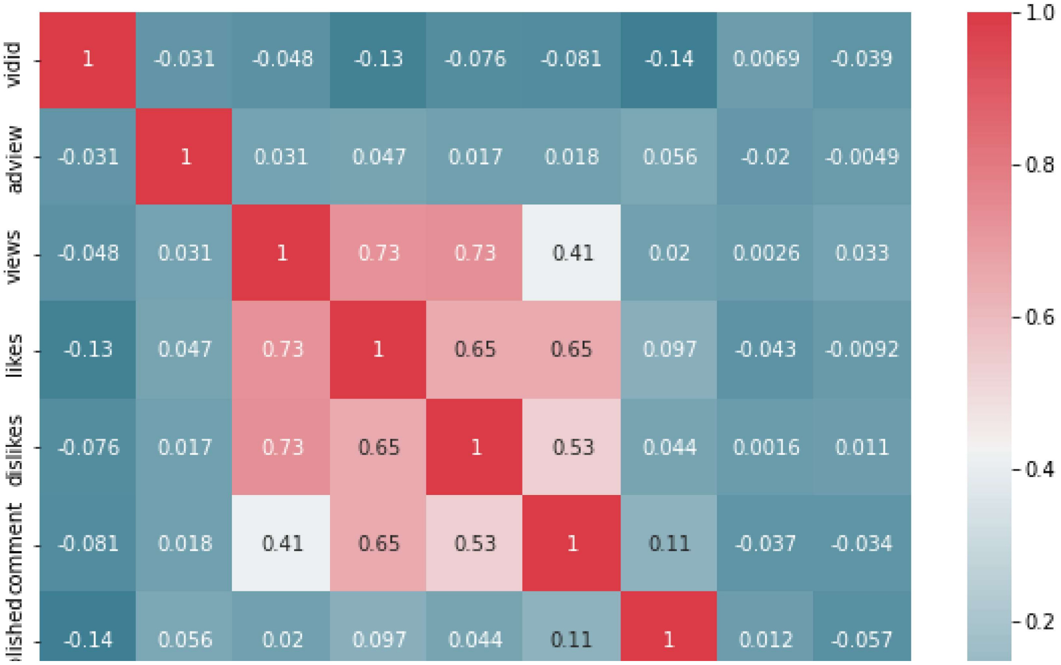
```
# Visualization
# Individual Plots
```

```
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(220, 16
square=True, ax=ax,annot=True)
plt.show()
```



	vidid	adview	views	likes	dislikes	comment	published	duration	category
vidid	1	-0.031	-0.048	-0.13	-0.076	-0.081	-0.14	0.0069	-0.039
adview	-0.031	1	0.031	0.047	0.017	0.018	0.056	-0.02	-0.0049
views	-0.048	0.031	1	0.73	0.73	0.41	0.02	0.0026	0.033
likes	-0.13	0.047	0.73	1	0.65	0.65	0.097	-0.043	-0.0092
dislikes	-0.076	0.017	0.73	0.65	1	0.53	0.044	0.0016	0.011
comment	-0.081	0.018	0.41	0.65	0.53	1	0.11	-0.037	-0.034
lished	-0.14	0.056	0.02	0.097	0.044	0.11	1	0.012	-0.057

Normalise your data and split the data into training, validation and test set in the appropriate ratio.

```
# 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=0.2, randc
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(11708, 7)
(2928, 7)
```

```
(11708, 1)
(2928, 1)
```

```
# 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
```

Use linear regression, support vector regressor, random forest and for training and get errors.

```
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
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:993: DataConversionW
  y = column_or_1d(y, warn=True)
Mean Absolute Error: 3707.378005824532
Mean Squared Error: 835663131.1210337
Root Mean Squared Error: 28907.83857573986
```

Use Decision Tree Regressor and Random Forest Regressors.

```
# 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: 2590.3876366120217
     Mean Squared Error: 882490897.8903688
     Root Mean Squared Error: 29706.748356061606
```

```
# 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_depth = max_depth, min
random_forest.fit(X_train,y_train)
print_error(X_test,y_test, random_forest)
```

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

     Mean Absolute Error: 3312.3881651457073
     Mean Squared Error: 626536110.9322995
     Root Mean Squared Error: 25030.703364713896
```

Build an artificial neural network and train it with different layers and hyperparameters.

```
# Artificial Neural Network
import tensorflow as tf
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=tf.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 [==============================] - 1s 2ms/step - loss: 767407616.0000 - mean_
Epoch 2/100
366/366 [==============================] - 1s 2ms/step - loss: 767406528.0000 - mean_
Epoch 3/100
366/366 [==============================] - 1s 2ms/step - loss: 767403072.0000 - mean_
Epoch 4/100
366/366 [==============================] - 1s 1ms/step - loss: 767395712.0000 - mean_
Epoch 5/100
366/366 [==============================] - 1s 2ms/step - loss: 767382976.0000 - mean_
Epoch 6/100
366/366 [==============================] - 1s 1ms/step - loss: 767365504.0000 - mean_
Epoch 7/100
366/366 [==============================] - 1s 1ms/step - loss: 767342976.0000 - mean_
Epoch 8/100
366/366 [==============================] - 1s 2ms/step - loss: 767314304.0000 - mean_
Epoch 9/100
366/366 [==============================] - 1s 2ms/step - loss: 767280384.0000 - mean_
Epoch 10/100
366/366 [==============================] - 1s 2ms/step - loss: 767239360.0000 - mean_
Epoch 11/100
366/366 [==============================] - 1s 2ms/step - loss: 767192256.0000 - mean_
Epoch 12/100
366/366 [==============================] - 1s 2ms/step - loss: 767140992.0000 - mean_
Epoch 13/100
366/366 [==============================] - 1s 2ms/step - loss: 767083200.0000 - mean_
Epoch 14/100
366/366 [==============================] - 1s 2ms/step - loss: 767020416.0000 - mean_
Epoch 15/100
366/366 [==============================] - 1s 2ms/step - loss: 766952768.0000 - mean_
Epoch 16/100
366/366 [==============================] - 1s 2ms/step - loss: 766879488.0000 - mean_
Epoch 17/100
366/366 [==============================] - 1s 2ms/step - loss: 766802880.0000 - mean_
Epoch 18/100
366/366 [==============================] - 1s 2ms/step - loss: 766721408.0000 - mean_
Epoch 19/100
366/366 [==============================] - 1s 2ms/step - loss: 766636864.0000 - mean_
Epoch 20/100
366/366 [==============================] - 1s 2ms/step - loss: 766549248.0000 - mean_
Epoch 21/100
366/366 [==============================] - 1s 2ms/step - loss: 766463232.0000 - mean_
Epoch 22/100
366/366 [==============================] - 1s 2ms/step - loss: 766374208.0000 - mean_
Epoch 23/100
366/366 [==============================] - 1s 2ms/step - loss: 766282048.0000 - mean_
Epoch 24/100
```

```
366/366 [==============================] - 1s 1ms/step - loss: 766186560.0000 - mean_
Epoch 25/100
366/366 [==============================] - 1s 2ms/step - loss: 766090240.0000 - mean_
Epoch 26/100
366/366 [==============================] - 1s 2ms/step - loss: 765996864.0000 - mean_
Epoch 27/100
366/366 [==============================] - 1s 2ms/step - loss: 765899392.0000 - mean_
Epoch 28/100
366/366 [==============================] - 1s 2ms/step - loss: 765802240.0000 - mean_
```

Save your model and predict on test set.

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

```
import joblib
classifer = joblib.load("decisiontree_youtubeadview.pkl")
```

```
prediction = classifer.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)))
```

```
Mean Absolute Error: 2590.3876366120217
Mean Squared Error: 882490897.8903688
Root Mean Squared Error: 29706.748356061606
```

Youtube adview Prediction Project

> Test_Data using test.csv

```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm
```

```
#loading the data
df = pd.read_csv( "test.csv" );
df.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

```
#mapping category feature to numbers
category = {'A':1,'B':2,'C':3,'D':4,'E':5,'F':6,'G':7,'H':8}

#transforming category
df['category'] = df['category'].map(category)
df.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

```
df.shape
```

```
(8764, 8)
```

```
# removing F from likes, dislikes, views, comments and adviews
df = df[df.views != 'F']
df = df[df.comment != 'F']
df = df[df.dislikes != 'F']
df = df[df.likes != 'F']
```

```
#rows with F in views, likes, dislikes and comment is removed
df.shape
```

```
(8549, 8)
```

```
df.dtypes
# as we can see likes, views etc has object dtypes. So, need to covert to numeric
```

```
vidid        object
```

```
views        object
likes        object
dislikes     object
comment      object
published    object
duration     object
category      int64
dtype: object
```

```
df['views'] = pd.to_numeric(df['views'])
df['comment'] = pd.to_numeric(df['comment'])
df['likes'] = pd.to_numeric(df['likes'])
df['dislikes'] = pd.to_numeric(df['dislikes'])
```

```
# storing vidid column in another variable for further use
col_vidid = df['vidid']
```

```
# performing encoding on vivid, duration and published
from sklearn.preprocessing import LabelEncoder
df['published'] = LabelEncoder().fit_transform(df['published'])
df['vidid'] = LabelEncoder().fit_transform(df['vidid'])
df['duration'] = LabelEncoder().fit_transform(df['duration'])
df.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"):
```

```
      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( "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)

df["duration"]=time1
df.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

```
df = df.drop(['vidid'],axis=1)
df.head()
```

	views	likes	dislikes	comment	published	duration	category
0	440238	6153	218	1377	2053	449	2

```
import joblib
classifer = joblib.load("decisiontree_youtubeadview.pkl")
prediction = classifer.predict(df)
prediction
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature
  f"X has feature names, but {self.__class__.__name__} was fitted without"
array([58., 58., 58., ..., 58., 58., 58.])
```

```
prediction
```

```
array([58., 58., 58., ..., 58., 58., 58.])
```

```
#saving prediction as csv file
np.savetxt('Predictions_Submission.csv',prediction,delimiter=',')
```

THANK YOU