# Project 2

# Analyze the dataset and train ml model

## Problem statement:

A streaming platform company allows different streamers to use its platform to conduct/deliver a live stream session. Now, the company wants to know which are its top streamers. But the company doesn't have a labeled dataset that indicates which are the top streamers and which are not. This dataset is generated from the live-streaming platform. It consists of all the streams that were conducted on the platform and the information related to each stream - its duration, country it was streamed from, no. of comments received during the stream, no. of viewers who attended the live stream etc.

## Tasks :

Analyze the dataset to come up with the top 20% streamers. 2. Label these top 20% streamers as "good" streamers, and the remaining as "bad" streamers. This will become your target variable. Now create a binary classification ML model that can classify whether any streamer is a good streamer or not.

### Tools / Skills Used :

1. Python Programming
2. Jupyter Notebook
3. Pandas
4. Numpy
5. Matplotlib
6. Seaborn
7. Excel
8. Data Visualization
9. Machine Learning

## Workflow :

## 1.Importing all the library.

```
%matplotlib inline
import numpy as np
import pandas as pd
from pandas.plotting import scatter_matrix
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
```

## 2.Loading the dataset in jupyter notebook :

```
all_stream_info = pd.read_csv("all_stream_info.csv")
all_stream_info_data_dict =  pd.read_csv("all_stream_info_data_dictionary.csv")
```

```
all_stream_info.head()
```

| | liveStreamID | beginTime | endTime | duration | closeBy | maxLiveViewerCount | maxLiveViewerTime | privateLiveStream | receivedLikeCount | streamerType | ... | i |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 109437538 | 2020-06-22 11:55:21 UTC | 2020-06-22 16:37:19 UTC | 16918 | normalEnd | 363 | 2020-06-22 16:28:17.87 UTC | 0 | 11092 | 0 | ... | |
| 1 | 109441785 | 2020-06-22 14:55:26 UTC | 2020-06-22 21:31:19 UTC | 23753 | normalEnd | 100 | 2020-06-22 19:07:52.872 UTC | 0 | 772 | 0 | ... | |
| 2 | 109438205 | 2020-06-22 12:20:34 UTC | 2020-06-22 16:02:46 UTC | 13332 | disconnect | 471 | 2020-06-22 14:53:26.692 UTC | 0 | 19403 | 0 | ... | |
| 3 | 109438917 | 2020-06-22 12:54:21 UTC | 2020-06-22 14:47:27 UTC | 6786 | normalEnd | 44 | 2020-06-22 14:29:13.806 UTC | 0 | 191 | 0 | ... | |
| 4 | 109442185 | 2020-06-22 15:18:20 UTC | 2020-06-22 15:48:02 UTC | 1782 | normalEnd | 52 | 2020-06-22 15:42:33.849 UTC | 0 | 77 | 0 | ... | |

5 rows × 24 columns

## 3.Now we will do some EDA on this dataset and we will handle all the missing value and categorical columns .

```
all_stream_info.describe()
```

| | liveStreamID | duration | maxLiveViewerCount | privateLiveStream | receivedLikeCount | streamerType | cultureGroup | isContracted | uniqueViewerCount |
|---|---|---|---|---|---|---|---|---|---|
| count | 3.148000e+03 | 3148.000000 | 3148.000000 | 3148.0 | 3148.000000 | 3148.0 | 0.0 | 0.0 | 3148.000000 |
| mean | 1.093797e+08 | 3495.864041 | 15.416773 | 0.0 | 358.428526 | 0.0 | NaN | NaN | 55.416773 |
| std | 3.381310e+04 | 6651.916426 | 45.505183 | 0.0 | 2145.579197 | 0.0 | NaN | NaN | 160.855894 |
| min | 1.093211e+08 | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.0 | NaN | NaN | 0.000000 |
| 25% | 1.093518e+08 | 136.750000 | 0.000000 | 0.0 | 0.000000 | 0.0 | NaN | NaN | 0.000000 |
| 50% | 1.093835e+08 | 1032.500000 | 2.000000 | 0.0 | 0.000000 | 0.0 | NaN | NaN | 2.000000 |
| 75% | 1.094082e+08 | 4801.500000 | 8.000000 | 0.0 | 62.000000 | 0.0 | NaN | NaN | 39.000000 |
| max | 1.094427e+08 | 121258.000000 | 493.000000 | 0.0 | 65831.000000 | 0.0 | NaN | NaN | 2385.000000 |

```
missing_categorical = [var for var in all_stream_info.columns if all_stream_info[var].isnull().mean()>0 and all_stream_info[var]
```

```
missing_categorical
```

```
['closeBy', 'maxLiveViewerTime']
```

```
missing_numerical = [var for var in all_stream_info.columns if all_stream_info[var].isnull().mean()>0 and all_stream_info[var].d
```


```
missing_categorical
```

```
['closeBy', 'maxLiveViewerTime']
```


```
missing_numerical = [var for var in all_stream_info.columns if all_stream_info[var].isnull().mean()>0 and all_stream_info[var].d
```

```
missing_numerical
```

```
['cultureGroup', 'isContracted', 'avgViewerDuration']
```

## Handling Categorical Values

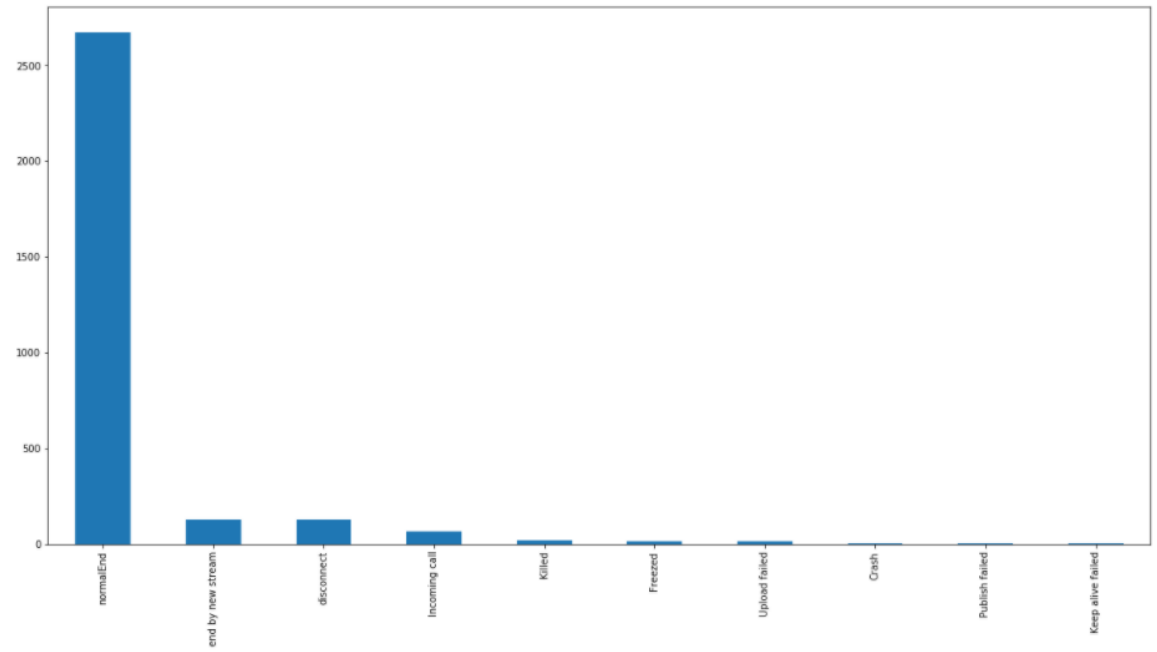
```
all_stream_info["closeBy"].isnull().mean()
```

```
0.030813214739517154
```

```
all_stream_info["closeBy"].isnull().sum()
```

```
97
```

```
plt.figure(figsize=(20,10))
all_stream_info["closeBy"].value_counts().plot.bar()
plt.show()
```



```
all_stream_info["closeBy"].mode()
```

```
0    normalEnd
dtype: object
```

```
all_stream_info["closeBy"].fillna('normalEnd',inplace = True)
```

```
all_stream_info["closeBy"].isnull().sum()
```

```
0
```

```
all_stream_info["maxLiveViewerTime"].isnull().mean()
```

```
0.4161372299872935
```

```
all_stream_info["maxLiveViewerTime"].isnull().sum()
```

```
1310
```

```
all_stream_info["maxLiveViewerTime"].mode()
```

```
0       2020-06-15 00:15:33.442 UTC
1       2020-06-15 00:21:43.015 UTC
2       2020-06-15 00:30:03.571 UTC
3       2020-06-15 00:32:51.494 UTC
4       2020-06-15 00:34:21.276 UTC
                   ...
1833    2020-06-22 16:45:19.952 UTC
1834    2020-06-22 17:04:12.298 UTC
1835    2020-06-22 17:19:39.697 UTC
1836    2020-06-22 17:19:48.148 UTC
1837    2020-06-22 19:07:52.872 UTC
Length: 1838, dtype: object
```

---

# Handling Numerical values

### 'cultureGroup', 'isContracted', 'avgViewerDuration' in these columns there is null values

```
all_stream_info["cultureGroup"].isnull().sum()
```

```
3148
```

```
all_stream_info["cultureGroup"].isnull().mean()
```

```
1.0
```

```
all_stream_info["isContracted"].isnull().sum()
```

```
3148
```

```
all_stream_info["isContracted"].isnull().mean()
```

```
1.0
```

```
#avgViewerDuration
all_stream_info["avgViewerDuration"].isnull().sum()
```

```
1459
```

```
all_stream_info["avgViewerDuration"].isnull().mean()
```

```
0.46346886912325286
```

```
all_stream_info["avgViewerDuration"].mode()
```

```
0    7.0
dtype: float64
```

---

```
all_stream_info["avgViewerDuration"].mean()
```

```
241.04946714031965
```

```
all_stream_info["avgViewerDuration"].fillna(7,inplace = True)
```

```
all_stream_info["avgViewerDuration"].isnull().sum()
```

```
0
```

```
all_stream_info.head()
```

| | liveStreamID | beginTime | endTime | duration | closeBy | maxLiveViewerCount | maxLiveViewerTime | privateLiveStream | receivedLikeCount | streamerType | ... | i |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 109437538 | 2020-06-22 11:55:21 UTC | 2020-06-22 16:37:19 UTC | 16918 | normalEnd | 363 | 2020-06-22 16:28:17.87 UTC | 0 | 11092 | 0 | ... | |
| 1 | 109441785 | 2020-06-22 14:55:26 UTC | 2020-06-22 21:31:19 UTC | 23753 | normalEnd | 100 | 2020-06-22 19:07:52.872 UTC | 0 | 772 | 0 | ... | |
| 2 | 109438205 | 2020-06-22 12:20:34 UTC | 2020-06-22 16:02:46 UTC | 13332 | disconnect | 471 | 2020-06-22 14:53:26.692 UTC | 0 | 19403 | 0 | ... | |
| 3 | 109438917 | 2020-06-22 12:54:21 UTC | 2020-06-22 14:47:27 UTC | 6786 | normalEnd | 44 | 2020-06-22 14:29:13.806 UTC | 0 | 191 | 0 | ... | |
| 4 | 109442185 | 2020-06-22 15:18:20 UTC | 2020-06-22 15:48:02 UTC | 1782 | normalEnd | 52 | 2020-06-22 15:42:33.849 UTC | 0 | 77 | 0 | ... | |

5 rows × 24 columns

```
all_stream_info.columns
```

```
Index(['liveStreamID', 'beginTime', 'endTime', 'duration', 'closeBy',
       'maxLiveViewerCount', 'maxLiveViewerTime', 'privateLiveStream',
       'receivedLikeCount', 'streamerType', 'isShow', 'cultureGroup', 'userID',
       'registerCountry', 'uniqueViewerCount', 'ios',
       'durationGTE2min', 'durationGTE10min',
       'avgViewerDuration', 'count', 'receivePointEstimated'],
      dtype='object')
```

lick to expand output; double click to hide output

```
all_stream_info.drop(['beginTime', 'endTime','maxLiveViewerTime','cultureGroup','isContracted'], axis = 1,inplace=True)
```

```
all_stream_info.head()
```

| | liveStreamID | duration | closeBy | maxLiveViewerCount | privateLiveStream | receivedLikeCount | streamerType | isShow | userID | registerCountry | unique |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 109437538 | 16918 | normalEnd | 363 | 0 | 11092 | 0 | False | 63a8b9eb-c4a6-4ff6-8aaa-5f4c6f6fd4d7 | United States | |
| 1 | 109441785 | 23753 | normalEnd | 100 | 0 | 772 | 0 | False | 6cd90016-b679-4a7b-8cc6-2c43d4590cca | United States | |
| 2 | 109438205 | 13332 | disconnect | 471 | 0 | 19403 | 0 | False | e4f04b19-ad3e-4c9d-7f29-6a2f6c93c842 | United States | |
| 3 | 109438917 | 6786 | normalEnd | 44 | 0 | 191 | 0 | False | b4dc876b-06fe-48b2-a40b-fa7f5007697a | United States | |
| 4 | 109442185 | 1782 | normalEnd | 52 | 0 | 77 | 0 | False | 8258d28e-47b9-4ce8-a765-29175b663a1d | United States | |

## 4.Exporting all the edits in csv file .

```
all_stream_info.to_csv("updated_csv.csv")
```

## 5. We have elliminate some columns which have no use in model building and saved in csv file now uploading that file in notebook and doing some EDA and moving towards model building .

```
data = pd.read_csv("updated_csv.csv")
```

```
data.head()
```

| erID | registerCountry | uniqueViewerCount | ios | android | durationGTE5sec | durationGTE2min | durationGTE10min | avgViewerDuration | count | receivePointEstimated |
|---|---|---|---|---|---|---|---|---|---|---|
| 6eb-4ff6-laaa-d4d7 | United States | 779 | 46 | 30 | 68 | 17 | 13 | 424.32 | 1108 | 1310 |
| f016-a7b-kcc6-0cca | United States | 821 | 54 | 38 | 68 | 20 | 8 | 186.28 | 2318 | 90701 |
| b19-fc9d-7f29-c842 | United States | 1605 | 529 | 223 | 696 | 60 | 17 | 81.14 | 1199 | 222376 |
| f76b-8b2-x40b-697a | United States | 96 | 23 | 24 | 41 | 12 | 8 | 389.91 | 432 | 4313 |
| f28e-ice8-i765-3a1d | United States | 109 | 20 | 18 | 38 | 17 | 5 | 222.76 | 162 | 1547 |

```
data['duration'].describe()
```

```
count      3148.000000
mean       3495.864041
std        6651.916426
min           0.000000
25%         136.750000
50%        1032.500000
75%        4801.500000
max      121258.000000
Name: duration, dtype: float64
```

```
data.describe()
```

| ype | uniqueViewerCount | ios | android | durationGTE5sec | durationGTE2min | durationGTE10min | avgViewerDuration | count | receivePointEstimate |
|---|---|---|---|---|---|---|---|---|---|
| 8.0 | 3148.000000 | 3148.000000 | 3148.000000 | 3148.000000 | 3148.000000 | 3148.000000 | 3148.000000 | 3148.000000 | 3.148000e+0 |
| 0.0 | 55.416773 | 26.285896 | 20.519377 | 44.560991 | 8.994917 | 3.937103 | 132.574825 | 126.772872 | 8.874376e+0 |
| 0.0 | 160.855894 | 81.280647 | 60.532012 | 131.896474 | 26.843270 | 13.057200 | 213.669542 | 330.297666 | 4.778547e+0 |
| 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000e+0 |
| 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 7.000000 | 0.000000 | 0.000000e+0 |
| 0.0 | 2.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 18.000000 | 1.000000 | 0.000000e+0 |
| 0.0 | 39.000000 | 20.000000 | 11.000000 | 31.000000 | 8.000000 | 3.000000 | 196.567500 | 112.250000 | 1.501500e+0 |
| 0.0 | 2385.000000 | 1639.000000 | 730.000000 | 2142.000000 | 493.000000 | 302.000000 | 2855.000000 | 5342.000000 | 1.474727e+0 |

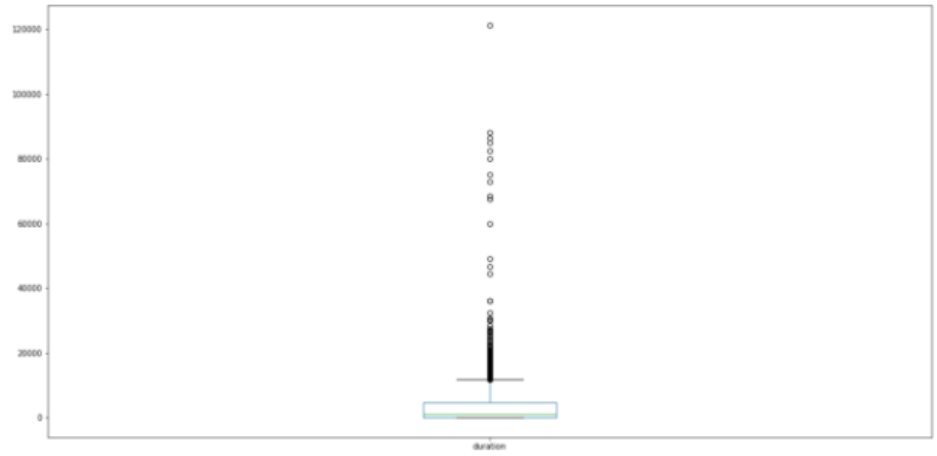
```
data.shape
```

```
(3148, 20)
```

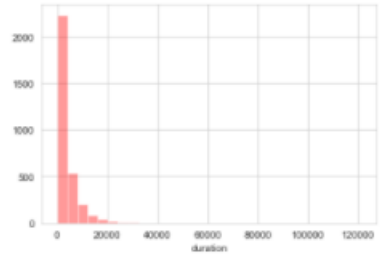
```
data.groupby('closeBy').size()
```

```
closeBy
Crash                   4
Freezed                16
Incoming call          65
Keep alive failed       1
Killed                 21
Publish failed          2
Upload failed          13
disconnect            127
end by new stream     129
normalEnd            2770
dtype: int64
```

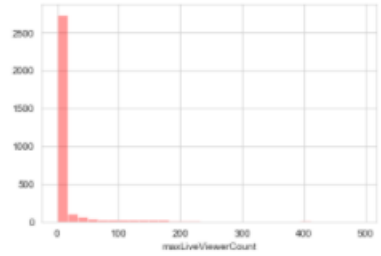
```
plt.figure(figsize=(20,10))
data["duration"].plot(kind='box',subplots=True,layout=(10,10),sharex=False,sharey=False)
plt.show()
```



```
sns.set_style('whitegrid')
sns.distplot(data["duration"], kde = False, color ='red', bins = 30)
plt.show()
```

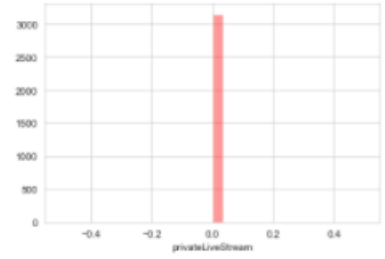



```
sns.set_style('whitegrid')
sns.distplot(data["maxLiveViewerCount"], kde = False, color ='red', bins = 30)
plt.show()
```

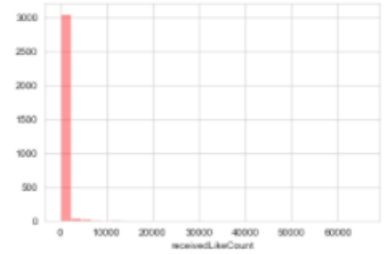



---

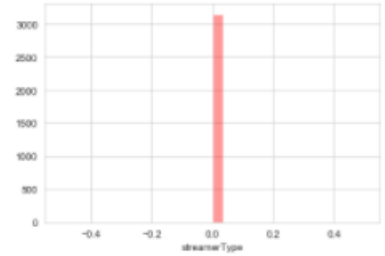
```
sns.set_style('whitegrid')
sns.distplot(data["privateLiveStream"], kde = False, color ='red', bins = 30)
plt.show()
```




```
sns.set_style('whitegrid')
sns.distplot(data["receivedLikeCount"], kde = False, color ='red', bins = 30)
plt.show()
```

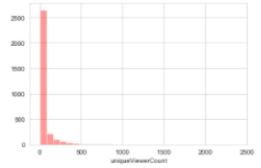



```
sns.set_style('whitegrid')
sns.distplot(data["streamerType"], kde = False, color ='red', bins = 30)
plt.show()
```



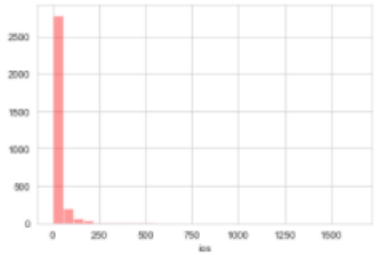

---

```
sns.set_style('whitegrid')
sns.distplot(data["uniqueViewerCount"], kde = False, color ='red', bins = 30)
plt.show()
```

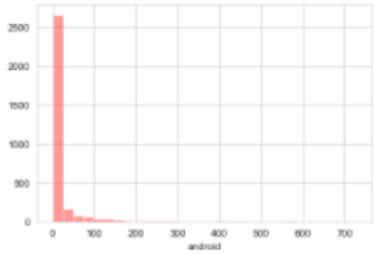



---

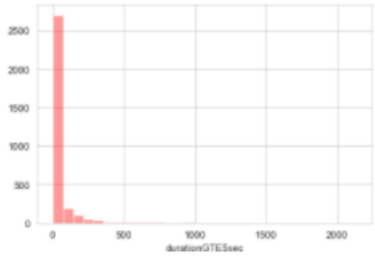
```
sns.set_style('whitegrid')
sns.distplot(data["ios"], kde = False, color ='red', bins = 30)
plt.show()
```




```
sns.set_style('whitegrid')
sns.distplot(data["android"], kde = False, color ='red', bins = 30)
plt.show()
```

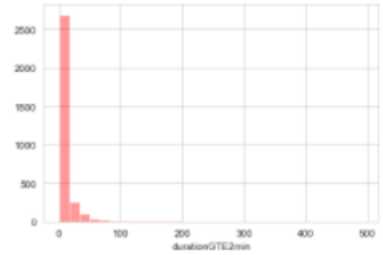



```
sns.set_style('whitegrid')
sns.distplot(data["durationGTE5sec"], kde = False, color ='red', bins = 30)
plt.show()
```

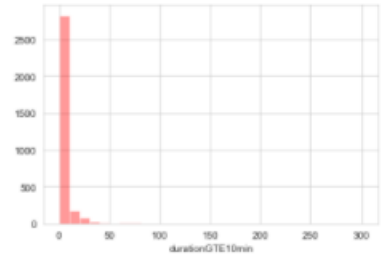



---

```
sns.set_style('whitegrid')
sns.distplot(data["durationGTE2min"], kde = False, color ='red', bins = 30)
plt.show()
```




```
sns.set_style('whitegrid')
sns.distplot(data["durationGTE10min"], kde = False, color ='red', bins = 30)
plt.show()
```




---

## 6.After Adding A Columns Of Top Streamer Uploading the data Set in notebook .

```
data1 = pd.read_csv('Newupdated_csv.csv')
```

```
data1.head()
```

| | Unnamed: 0 | liveStreamID | duration | closeBy | maxLiveViewerCount | privateLiveStream | receivedLikeCount | streamerType | isShow | userID | ... | u |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 109437538 | 16918 | normalEnd | 363 | 0 | 11092 | 0 | False | 63a8b9eb-c4a6-4ff6-8aaa-5f4c6f6fd4d7 | ... | |
| 1 | 1 | 109441785 | 23753 | normalEnd | 100 | 0 | 772 | 0 | False | 6cd90016-b679-4a7b-8cc6-2c43d4590cca | ... | |
| 2 | 2 | 109438205 | 13332 | disconnect | 471 | 0 | 19403 | 0 | False | e4f04b19-ad3e-4c9d-7f29-6a2f6c93c842 | ... | |
| 3 | 3 | 109438917 | 6786 | normalEnd | 44 | 0 | 191 | 0 | False | b4dc876b-06fe-48b2-a40b- | ... | |

## Droping all the columns which we are not going to use in our model building .

```
#df.drop(['A'], axis = 1)
```

```
data1.drop(['Unnamed: 0',"liveStreamID",'userID'],axis = 1,inplace=True)
```

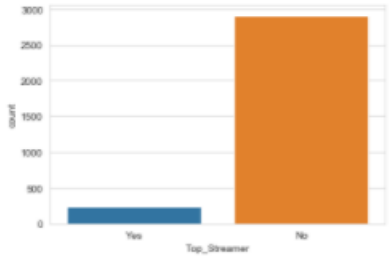
```
data1.head(20)
```

| | duration | closeBy | maxLiveViewerCount | privateLiveStream | receivedLikeCount | streamerType | isShow | registerCountry | uniqueViewerCount | ios | androi |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16918 | normalEnd | 363 | 0 | 11092 | 0 | False | United States | 779 | 46 | 3 |
| 1 | 23753 | normalEnd | 100 | 0 | 772 | 0 | False | United States | 821 | 54 | 3 |
| 2 | 13332 | disconnect | 471 | 0 | 19403 | 0 | False | United States | 1605 | 529 | 22 |
| 3 | 6786 | normalEnd | 44 | 0 | 191 | 0 | False | United States | 96 | 23 | 2 |
| 4 | 1782 | normalEnd | 52 | 0 | 77 | 0 | False | United States | 109 | 20 | 1 |
| 5 | 1749 | normalEnd | 45 | 0 | 107 | 0 | False | United States | 116 | 25 | 2 |
| 6 | 1756 | normalEnd | 50 | 0 | 199 | 0 | False | United States | 120 | 23 | 1 |
| 7 | 1792 | normalEnd | 365 | 0 | 3424 | 0 | False | United States | 377 | 83 | 5 |
| 8 | 233 | Incoming call | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 9 | 5264 | end by new stream | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 10 | 102 | normalEnd | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 11 | 307 | Incoming call | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 12 | 19 | normalEnd | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 13 | 245 | normalEnd | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 14 | 4 | normalEnd | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 15 | 8 | normalEnd | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 16 | 96 | normalEnd | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 17 | 3 | normalEnd | 0 | 0 | 0 | 0 | False | United States | 0 | 0 | |
| 18 | 5959 | normalEnd | 17 | 0 | 380 | 0 | False | United States | 86 | 0 | |
| 19 | 169 | normalEnd | 8 | 0 | 11 | 0 | False | United States | 8 | 1 | |

```
var1 = data1.groupby('Top_Streamer')
var1.mean()
```

| Top_Streamer | duration | maxLiveViewerCount | privateLiveStream | receivedLikeCount | streamerType | isShow | uniqueViewerCount | ios | android | c |
|---|---|---|---|---|---|---|---|---|---|---|
| No | 2844.738742 | 12.724991 | 0.0 | 290.635957 | 0.0 | False | 39.936061 | 17.885528 | 15.333104 | |
| Yes | 11421.066946 | 48.179916 | 0.0 | 1183.569038 | 0.0 | False | 243.841004 | 126.531381 | 83.644351 | |

```
sns.countplot(x=data1.Top_Streamer ,data = data1)
plt.show()
```




```
obj_col = []
num_col = []
for col in data1.columns:
    if data1[col].dtype=='O':
        obj_col.append(col)
    else:
        num_col.append(col)
```

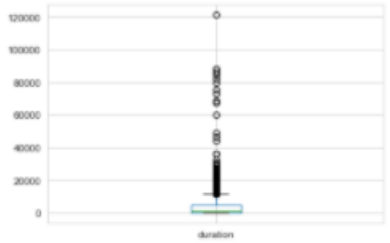
```
num_col
```

```
['duration',
 'maxLiveViewerCount',
 'privateLiveStream',
 'receivedLikeCount',
 'streamerType',
 'isShow',
 'uniqueViewerCount',
 'ios',
 'android',
 'durationGTE5sec',
 'durationGTE2min',
 'durationGTE10min',
 'avgViewerDuration',
 'count',
 'receivePointEstimated']
```

```
obj_col
```

```
['closeBy', 'registerCountry', 'Top_Streamer']
```

```
data1.boxplot(column=['duration'])
plt.show()
```




## 7.Data Pre-processing For ML Model Building .

```
from sklearn.preprocessing import LabelEncoder
labelEncoder_X = LabelEncoder()
data1['closeBy'] = labelEncoder_X.fit_transform(data1['closeBy'])
data1['registerCountry'] = labelEncoder_X.fit_transform(data1['registerCountry'])
data1['Top_Streamer'] = labelEncoder_X.fit_transform(data1['Top_Streamer'])
```

```
data1.head()
```

| erCountry | uniqueViewerCount | ios | android | durationGTE5sec | durationGTE2min | durationGTE10min | avgViewerDuration | count | receivePointEstimated | Top_Strea |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 779 | 46 | 30 | 68 | 17 | 13 | 424.32 | 1108 | 13105 | |
| 0 | 821 | 54 | 38 | 68 | 20 | 8 | 186.28 | 2318 | 90701 | |
| 0 | 1605 | 529 | 223 | 696 | 60 | 17 | 81.14 | 1199 | 222376 | |
| 0 | 96 | 23 | 24 | 41 | 12 | 8 | 389.91 | 432 | 4313 | |
| 0 | 109 | 20 | 18 | 38 | 17 | 5 | 222.76 | 162 | 1547 | |

```
#Attriton is dependent var
from sklearn.preprocessing import LabelEncoder
label_encoder_y=LabelEncoder()
data1['Top_Streamer']=label_encoder_y.fit_transform(data1['Top_Streamer'])
```

```
data1.head()
```

| erCountry | uniqueViewerCount | ios | android | durationGTE5sec | durationGTE2min | durationGTE10min | avgViewerDuration | count | receivePointEstimated | Top_Strea |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 779 | 46 | 30 | 68 | 17 | 13 | 424.32 | 1108 | 13105 | |
| 0 | 821 | 54 | 38 | 68 | 20 | 8 | 186.28 | 2318 | 90701 | |
| 0 | 1605 | 529 | 223 | 696 | 60 | 17 | 81.14 | 1199 | 222376 | |
| 0 | 96 | 23 | 24 | 41 | 12 | 8 | 389.91 | 432 | 4313 | |
| 0 | 109 | 20 | 18 | 38 | 17 | 5 | 222.76 | 162 | 1547 | |

```
data1.columns
```

```
Index(['duration', 'closeBy', 'maxLiveViewerCount', 'privateLiveStream',
       'receivedLikeCount', 'streamerType', 'isShow', 'registerCountry',
       'uniqueViewerCount', 'ios', 'android', 'durationGTE5sec',
       'durationGTE2min', 'durationGTE10min', 'avgViewerDuration', 'count',
       'receivePointEstimated', 'Top_Streamer'],
      dtype='object')
```

```
corr_cols = data1[['duration', 'closeBy', 'maxLiveViewerCount', 'privateLiveStream',
       'receivedLikeCount', 'streamerType', 'isShow', 'registerCountry',
       'uniqueViewerCount', 'ios', 'android', 'durationGTE5sec',
       'durationGTE2min', 'durationGTE10min', 'avgViewerDuration', 'count',
       'receivePointEstimated', 'Top_Streamer' ]]
```

```
corr = corr_cols.corr()
plt.figure(figsize=(18,10))
sns.heatmap(corr, annot = True)
plt.show()
```

```
X = data1.iloc[:,:-1]
y = data1.iloc[:,-1]
```

```
X.head()
```

| isShow | registerCountry | uniqueViewerCount | ios | android | durationGTE5sec | durationGTE2min | durationGTE10min | avgViewerDuration | count | receivePointEstim |
|---|---|---|---|---|---|---|---|---|---|---|
| False | 0 | 779 | 46 | 30 | 68 | 17 | 13 | 424.32 | 1108 | 1 |
| False | 0 | 821 | 54 | 38 | 68 | 20 | 8 | 186.28 | 2318 | 9 |
| False | 0 | 1605 | 529 | 223 | 696 | 60 | 17 | 81.14 | 1199 | 22 |
| False | 0 | 96 | 23 | 24 | 41 | 12 | 8 | 389.91 | 432 | |
| False | 0 | 109 | 20 | 18 | 38 | 17 | 5 | 222.76 | 162 | |

```
y.head()
```

```
0    1
1    0
2    0
3    1
4    0
Name: Top_Streamer, dtype: int64
```

```
from sklearn.preprocessing import scale
X = scale(X)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.20,random_state = 30)
```

## 8.Training Ml Model .

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier()
```

```
model.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators wil
l change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', 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, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
predict = model.predict(X_test)
```

# 9. Model Evaluation .

## Model Evaluation

```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
accuracy_score(y_test,predict)
```

```
0.973015873015873
```

```
from pprint import pprint# Look at parameters used by our current forest
print('Parameters currently in use:\n')
pprint(model.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 '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,
 'n_estimators': 10,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

```
from sklearn.metrics import fbeta_score
```

```
print('F2 Score:', fbeta_score(y_test,predict, beta=2, average='micro'))
```

```
F2 Score: 0.973015873015873
```