Introduction

My decision was based off the fact that I am a gamer and I have recently found out that watching other gamers play their games is very interesting

In [33]:

```
import pandas as pd
```

In [34]:

```
data= pd.read_csv('twitchdata-update.csv', header='infer')
```

In [35]:

```
data
```

Out[35]:

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	Views gained	Partnered	Mature	La
0	xQcOW	6196161750	215250	222720	27716	3246298	1734810	93036735	True	False	
1	summit1g	6091677300	211845	310998	25610	5310163	1370184	89705964	True	False	
2	Gaules	5644590915	515280	387315	10976	1767635	1023779	102611607	True	True	Port
3	ESL_CSGO	3970318140	517740	300575	7714	3944850	703986	106546942	True	False	
4	Tfue	3671000070	123660	285644	29602	8938903	2068424	78998587	True	False	
...	...	...	...	...	...	...	...	...	...	...	
995	LITkillah	122524635	13560	21359	9104	601927	562691	2162107	True	False	S
996	빅헤드 (bighead033)	122523705	153000	3940	793	213212	52289	4399897	True	False	
997	마스카 (newmasca)	122452320	217410	6431	567	109068	-4942	3417970	True	False	
998	AndyMilonakis	122311065	104745	10543	1153	547446	109111	3926918	True	False	
999	Remx	122192850	99180	13788	1205	178553	59432	2049420	True	False	

1000 rows × 11 columns

In [36]:

```
data['Language'].unique()
```

Out[36]:

```
array(['English', 'Portuguese', 'Spanish', 'German', 'Korean', 'French',
       'Russian', 'Japanese', 'Chinese', 'Czech', 'Turkish', 'Italian',
       'Polish', 'Thai', 'Arabic', 'Slovak', 'Other', 'Hungarian',
       'Greek', 'Finnish', 'Swedish'], dtype=object)
```

In [37]:

```
data['Language'] = data['Language'].replace(['Portuguese', 'Spanish', 'German', 'Korean'
, 'French',
       'Russian', 'Japanese', 'Chinese', 'Czech', 'Turkish', 'Italian',
       'Polish', 'Thai', 'Arabic', 'Slovak', 'Other', 'Hungarian',
       'Greek', 'Finnish', 'Swedish'],'Non-English')
```

In [38]:

```
data['Mature'] = data['Mature'].replace([True], 1)
```

In [39]:

```
data['Mature'] = data['Mature'].replace([False], 0)
```

In [40]:

```
data['Partnered'] = data['Partnered'].replace([True], 1)
```

In [41]:

```
data['Partnered'] = data['Partnered'].replace([False], 0)
```

In [42]:

```
data['Language'] = data['Language'].replace(['English'], 1)
```

In [43]:

```
data['Language'] = data['Language'].replace(['Non-English'], 0)
```

In [44]:

```
data['Followers'].where(data['Followers'] > 1000000, 0, inplace=True)
```

In [45]:

```
data['Followers'].where(data['Followers'] <= 1000000, 1, inplace=True)
```

In [46]:

```
data['Peak viewers'].where(data['Peak viewers'] <= 100000, 1, inplace=True)
```

In [47]:

```
data['Average viewers'].where(data['Average viewers'] > 10000, 0, inplace=True)
```

In [48]:

```
data['Average viewers'].where(data['Average viewers'] <= 10000, 1, inplace=True)
```

In [49]:

```
data
```

Out[49]:

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	Views gained	Partnered	Mature	Lang
0	xQcOW	6196161750	215250	1	1	1	1734810	93036735	1	0	
1	summit1g	6091677300	211845	1	1	1	1370184	89705964	1	0	
2	Gaules	5644590915	515280	1	1	1	1023779	102611607	1	1	
3	ESL_CSGO	3970318140	517740	1	0	1	703986	106546942	1	0	
4	Tfue	3671000070	123660	1	1	1	2068424	78998587	1	0	
...	...	...	...	...	...	...	...	...	...	...	
995	LITkillah	122524635	13560	21359	0	0	562691	2162107	1	0	
996	빅헤드 (bighead033)	122523705	153000	3940	0	0	52289	4399897	1	0	
997	마스카 (newmasca)	122452320	217410	6431	0	0	-4942	3417970	1	0	
998	AndyMilonakis	122311065	104745	10543	0	0	109111	3926918	1	0	
999	Remx	122192850	99180	13788	0	0	59432	2049420	1	0	

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	Views gained	Partnered	Mature	Lang

1000 rows × 11 columns

Building the Decision Tree Classifier

In [50]:

```
from sklearn import tree
```

In [51]:

```
y = data['Followers']
```

In [52]:

```
y
```

Out[52]:

```
0      1
1      1
2      1
3      1
4      1
      ..
995    0
996    0
997    0
998    0
999    0
Name: Followers, Length: 1000, dtype: int64
```

In [53]:

```
X = data.drop(['Channel','Watch time(Minutes)','Stream time(minutes)','Followers','Follo
wers gained','Views gained','Followers gained','Views gained','Partnered'], axis = 1)
```

In [54]:

```
X
```

Out[54]:

	Peak viewers	Average viewers	Mature	Language
0	1	1	0	1
1	1	1	0	1
2	1	1	1	0
3	1	0	0	1
4	1	1	0	1
...	...	...	...	...
995	21359	0	0	0
996	3940	0	0	0
997	6431	0	0	0
998	10543	0	0	1
999	13788	0	0	0

1000 rows × 4 columns

In [55]:

```
type(y)
```

Out[55]:

```
pandas.core.series.Series
```

In [56]:

```
type(X)
```

Out[56]:

```
pandas.core.frame.DataFrame
```

In [57]:

```
clf = tree.DecisionTreeClassifier(criterion = 'entropy', max_depth = 3)
```

In [58]:

```
clf
```

Out[58]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [59]:

```
clf = clf.fit(X,y)
```

Plot the results of the tree

In [60]:

```
import pydotplus
```

In [61]:

```
from IPython.display import Image
```

In [62]:

```
dot_data = tree.export_graphviz(clf, feature_names=X.columns, class_names=['1M Followers'
,'Less than 1M'], filled=True, out_file=None)
```

In [63]:

```
graph = pydotplus.graph_from_dot_data(dot_data)
```

In [158]:

```
Image(graph.create_png())
```

```
---------------------------------------------------------------------------
InvocationException                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_22976/3935282027.py in <module>
----> 1 Image(graph.create_png())

C:\Python39\lib\site-packages\pydotplus\graphviz.py in <lambda>(f, prog)
   1795             self.__setattr__(
   1796                 'create_' + frmt,
-> 1797                 lambda f=frmt, prog=self.prog: self.create(format=f, prog=prog)
   1798             )
   1799             f = self.__dict__['create_' + frmt]

C:\Python39\lib\site-packages\pydotplus\graphviz.py in create(self, prog, format)
   1957             self.progs = find_graphviz()
   1958             if self.progs is None:
-> 1959                 raise InvocationException(
   1960                     'GraphViz\'s executables not found')
   1961
```

```
InvocationException: GraphViz's executables not found
```

Using the Decision Tree to Classify new Datasets

In [65]:

```
testData = [['Dalinar', 1,1,0,1,'1M Followers'], ['Kelek',0,1,1,1,'Less than 1M'], ['Wax
illium', 1,0,0,0,'Less than 1M'], ['Marasi',0,1,0,0,'1M Followers']]
```

In [66]:

```
testData
```

Out[66]:

```
[['Dalinar', 1, 1, 0, 1, '1M Followers'],
 ['Kelek', 0, 1, 1, 1, 'Less than 1M'],
 ['Waxillium', 1, 0, 0, 0, 'Less than 1M'],
 ['Marasi', 0, 1, 0, 0, '1M Followers']]
```

In [67]:

```
testData=pd.DataFrame(testData, columns=['Name','Peak viewers','Average viewers','Mature'
,'Language','Followers'])
```

In [68]:

```
testData
```

Out[68]:

	Name	Peak viewers	Average viewers	Mature	Language	Followers
0	Dalinar	1	1	0	1	1M Followers
1	Kelek	0	1	1	1	Less than 1M
2	Waxillium	1	0	0	0	Less than 1M
3	Marasi	0	1	0	0	1M Followers

In [69]:

```
testY=testData['Followers']
```

In [70]:

```
testY
```

Out[70]:

```
0     1M Followers
1     Less than 1M
2     Less than 1M
3     1M Followers
Name: Followers, dtype: object
```

In [71]:

```
testX=testData.drop(['Name', 'Followers'], axis=1)
```

In [72]:

```
testX
```

Out[72]:

	Peak viewers	Average viewers	Mature	Language
0	1	1	0	1

	Peak viewers	Average viewers	Mature	Language
1	0	1	1	1
2	1	0	0	0
3	0	1	0	0

In [73]:

```
predY=clf.predict(testX)
```

In [74]:

```
type(predY)
```

Out[74]:

```
numpy.ndarray
```

In [75]:

```
predY
```

Out[75]:

```
array([1, 1, 0, 1], dtype=int64)
```

In [76]:

```
predictions=pd.concat([testData['Name'],testData['Followers'], pd.Series(predY, name='Pr
edicted Followers')], axis=1)
```

In [77]:

```
predictions
```

Out[77]:

	Name	Followers	Predicted Followers
0	Dalinar	1M Followers	1
1	Kelek	Less than 1M	1
2	Waxillium	Less than 1M	0
3	Marasi	1M Followers	1

In [78]:

```
from sklearn.metrics import accuracy_score
import numpy as np
```

In [79]:

```
predY
```

Out[79]:

```
array([1, 1, 0, 1], dtype=int64)
```

In [81]:

```
predY = np.select([predY==1, predY==0], ["1M Followers", "Less than 1M"], predY)
```

In [80]:

```
predY
```

Out[80]:

```
array([1, 1, 0, 1], dtype=int64)
```

In [82]:

```
print('Accuracy on data is %.2f'%(accuracy_score(testY,predY)*100.))
```

```
Accuracy on data is 75.00
```

Logistic Regression

Filter unnecessary warnings

In [83]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [84]:

```
import numpy as np
import pandas as pd
```

Seeding

In [85]:

```
np.random.seed(5)
```

Read the Data

In [86]:

```
lgdata = pd.read_csv('twitchdata-update.csv', header = 'infer')
```

In [87]:

```
lgdata.head(20).T
```

Out[87]:

	0	1	2	3	4	5	6	7	
Channel	xQcOW	summit1g	Gaules	ESL_CSGO	Tfue	Asmongold	NICKMERCS	Fextralife	lolty
Watch time(Minutes)	6196161750	6091677300	5644590915	3970318140	3671000070	3668799075	3360675195	3301867485	2928356
Stream time(minutes)	215250	211845	515280	517740	123660	82260	136275	147885	122
Peak viewers	222720	310998	387315	300575	285644	263720	115633	68795	89
Average viewers	27716	25610	10976	7714	29602	42414	24181	18985	22
Followers	3246298	5310163	1767635	3944850	8938903	1563438	4074287	508816	3530
Followers gained	1734810	1370184	1023779	703986	2068424	554201	1089824	425468	951
Views gained	93036735	89705964	102611607	106546942	78998587	61715781	46084211	670137548	51349
Partnered	True	True	True	True	True	True	True	True	T
Mature	False	False	True	False	False	False	False	False	Fa
Language	English	English	Portuguese	English	English	English	English	English	Eng

In [88]:

```
lgdata.tail(30).T
```

Out[88]:

	970	971	972	973	974	975	976	977	
Channel	mailand	Buozzi	HeyarTV	LenaGol0vach	Sinner666	TommyKayLIVE	TMemoryy	Patriota	근 (runner(
Watch time(Minutes)	124936395	124926540	124878165	124876665	124812240	124680810	124620795	124595820	12452
Stream time(minutes)	86865	142785	155700	30210	24765	108105	130440	64305	5
Peak viewers	6867	4036	2343	9115	27996	3536	3244	29435	1
Average viewers	1400	872	786	4159	4530	1124	924	1951	
Followers	86561	75290	124479	274237	7102	127770	171107	646758	16
Followers gained	31605	39547	17695	40915	7101	58673	64750	344213	5
Views gained	3923343	5126434	2506556	2969471	21322548	2561302	1027960	3707623	427
Partnered	True	True	True	False	False	True	True	True	I
Mature	True	True	True	False	False	True	False	False	I
Language	German	Portuguese	French	Russian	Russian	English	English	Portuguese	Kc

11 rows × 30 columns

In [89]:

```
lgdata.sample(15).T
```

Out[89]:

	544	515	193	11	279	653	643	763	198	
Channel	bebe872	Sh4dowehhh	接接 (godjj)	LIRIK	shongxbong	Pengu	Broeki1	tebtv	울프 (lol_woolf)	F
Watch time(Minutes)	217343400	229762950	544706325	2832930285	400635750	182358345	186562710	161468685	532969650	
Stream time(minutes)	212205	112875	126705	128490	50310	70665	141675	69315	50910	
Peak viewers	4965	11788	12461	89170	94869	7707	13091	13639	73800	
Average viewers	1087	2012	4378	21739	7656	2452	1317	2177	9633	
Followers	70232	84234	331744	2666382	845158	775987	150944	76050	308528	
Followers gained	51251	70619	19325	199077	671127	241309	25995	59909	141297	
Views gained	2383295	6477747	20264128	50504526	4198232	5240516	4702541	10310607	9294132	
Partnered	True	True	True	True	True	True	True	True	True	
Mature	False	False	False	False	False	True	True	False	False	
Language	English	Russian	Chinese	English	Arabic	English	German	English	Korean	

In [90]:

```
lgdata.shape
```

Out[90]:

```
(1000, 11)
```

In [91]:

```
lgdata.columns
```

Out[91]:

```
Index(['Channel', 'Watch time(Minutes)', 'Stream time(minutes)',
       'Peak viewers', 'Average viewers', 'Followers', 'Followers gained',
       'Views gained', 'Partnered', 'Mature', 'Language'],
      dtype='object')
```

In [92]:

```
lgdata
```

Out[92]:

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	Views gained	Partnered	Mature	La
0	xQcOW	6196161750	215250	222720	27716	3246298	1734810	93036735	True	False	
1	summit1g	6091677300	211845	310998	25610	5310163	1370184	89705964	True	False	
2	Gaules	5644590915	515280	387315	10976	1767635	1023779	102611607	True	True	Port
3	ESL_CSGO	3970318140	517740	300575	7714	3944850	703986	106546942	True	False	
4	Tfue	3671000070	123660	285644	29602	8938903	2068424	78998587	True	False	
...	...	...	...	...	...	...	...	...	...	...	
995	LITkillah	122524635	13560	21359	9104	601927	562691	2162107	True	False	S
996	빅헤드 (bighead033)	122523705	153000	3940	793	213212	52289	4399897	True	False	
997	마스카 (newmasca)	122452320	217410	6431	567	109068	-4942	3417970	True	False	
998	AndyMilonakis	122311065	104745	10543	1153	547446	109111	3926918	True	False	
999	Remx	122192850	99180	13788	1205	178553	59432	2049420	True	False	

1000 rows × 11 columns

In [93]:

```
lgdata['Language'].unique()
```

Out[93]:

```
array(['English', 'Portuguese', 'Spanish', 'German', 'Korean', 'French',
       'Russian', 'Japanese', 'Chinese', 'Czech', 'Turkish', 'Italian',
       'Polish', 'Thai', 'Arabic', 'Slovak', 'Other', 'Hungarian',
       'Greek', 'Finnish', 'Swedish'], dtype=object)
```

In [94]:

```
lgdata['Language'] = lgdata['Language'].replace(['Portuguese', 'Spanish', 'German', 'Kor
ean', 'French',
        'Russian', 'Japanese', 'Chinese', 'Czech', 'Turkish', 'Italian',
        'Polish', 'Thai', 'Arabic', 'Slovak', 'Other', 'Hungarian',
        'Greek', 'Finnish', 'Swedish'], 'Non-English')
```

The value 'Non-English' represents any language that isn't English, that is, it includes Portuguese, Spanish, German, Korean, French, Russian, Japanese, Chinese, Czech, Turkish, Italian, Polish, Thai, Arabic, Slovak, Other, Hungarian, Greek, Finnish and Swedish

In [95]:

```
lgdata
```

```
lgdata
```

Out[95]:

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	Views gained	Partnered	Mature	Lang
0	xQcOW	6196161750	215250	222720	27716	3246298	1734810	93036735	True	False	E
1	summit1g	6091677300	211845	310998	25610	5310163	1370184	89705964	True	False	E
2	Gaules	5644590915	515280	387315	10976	1767635	1023779	102611607	True	True	E
3	ESL_CSGO	3970318140	517740	300575	7714	3944850	703986	106546942	True	False	E
4	Tfue	3671000070	123660	285644	29602	8938903	2068424	78998587	True	False	E
...	...	...	...	...	...	...	...	...	...	...	
995	LITkillah	122524635	13560	21359	9104	601927	562691	2162107	True	False	E
996	빅헤드 (bighead033)	122523705	153000	3940	793	213212	52289	4399897	True	False	E
997	마스카 (newmasca)	122452320	217410	6431	567	109068	-4942	3417970	True	False	E
998	AndyMilonakis	122311065	104745	10543	1153	547446	109111	3926918	True	False	E
999	Remx	122192850	99180	13788	1205	178553	59432	2049420	True	False	E

1000 rows × 11 columns

In [96]:

```
from collections import Counter
```

In [97]:

```
classes = Counter(lgdata['Language'].values)
```

In [98]:

```
classes
```

Out[98]:

```
Counter({'English': 485, 'Non-English': 515})
```

Create a dataframe to display the results

In [99]:

```
class_dist = pd.DataFrame(classes.most_common(), columns=['Class','Num_Observations'])
```

In [100]:

```
class_dist
```

Out[100]:

	Class	Num_Observations
0	Non-English	515
1	English	485

In [101]:

```
import matplotlib.pyplot as plt
```

Plots that are to appear in the notebook

In [102]:

```
%matplotlib inline
```

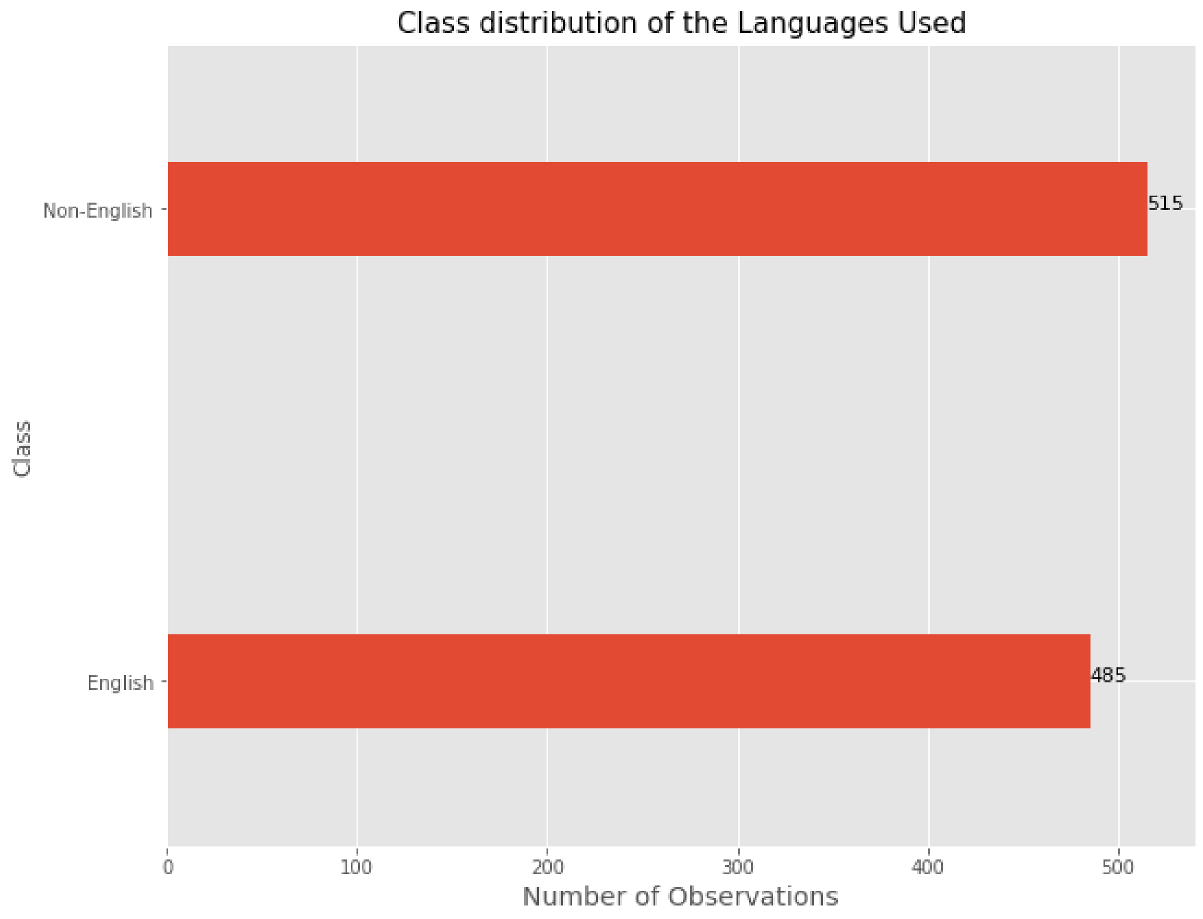
Stylizing the Plot

In [103]:

```
plt.style.use('ggplot')
```

In [104]:

```
subplot = class_dist.groupby('Class')['Num_Observations'].sum().plot(kind='barh', width=
0.2, figsize=(10,8))
subplot.set_title('Class distribution of the Languages Used', fontsize = 15)
subplot.set_xlabel('Number of Observations', fontsize = 14)
for i in subplot.patches:
    subplot.text(i.get_width() + 0.1, i.get_y() + 0.1, str(i.get_width()), fontsize=11)
```



MIssing values are likely to create issues later on

In [105]:

```
lgdata.describe().T
```

Out[105]:

	count	mean	std	min	25%	50%	75%	max
Watch time(Minutes)	1000.0	4.184279e+08	5.496355e+08	122192850.0	1.631899e+08	234990787.5	4.337399e+08	6.196162e+09
Stream								

	count	mean	std	min	25%	50%	75%	max
time(minutes)	1000.0	1.205152e+05	8.537620e+04	3465.0	7.375875e+04	108240.0	1.418438e+05	5.214450e+05
Peak viewers	1000.0	3.706505e+04	6.031431e+04	496.0	9.113750e+03	16676.0	3.756975e+04	6.393750e+05
Average viewers	1000.0	4.781040e+03	8.453685e+03	235.0	1.457750e+03	2425.0	4.786250e+03	1.476430e+05
Followers	1000.0	5.700541e+05	8.044134e+05	3660.0	1.705462e+05	318063.0	6.243322e+05	8.938903e+06
Followers gained	1000.0	2.055185e+05	3.399137e+05	-15772.0	4.375825e+04	98352.0	2.361308e+05	3.966525e+06
Views gained	1000.0	1.166817e+07	2.490572e+07	175788.0	3.880602e+06	6456323.5	1.219676e+07	6.701375e+08

In [106]:

```
lgdata.info
```

Out[106]:

```
<bound method DataFrame.info of                Channel  Watch time(Minutes)  Stream time(m
inutes)  \
0              xQcOW           6196161750               215250
1           summit1g           6091677300               211845
2             Gaules           5644590915               515280
3           ESL_CSGO           3970318140               517740
4               Tfue           3671000070               123660
..               ...                  ...                  ...
995        LITkillah            122524635                13560
996   빅헤드 (bighead033)            122523705                153000
997     마스카 (newmasca)            122452320                217410
998      AndyMilonakis            122311065               104745
999               Remx            122192850                99180

     Peak viewers  Average viewers  Followers  Followers gained  Views gained  \
0          222720            27716    3246298           1734810      93036735
1          310998            25610    5310163           1370184      89705964
2          387315            10976    1767635           1023779     102611607
3          300575             7714    3944850            703986     106546942
4          285644            29602    8938903           2068424      78998587
..            ...              ...        ...               ...           ...
995         21359             9104     601927            562691       2162107
996          3940              793     213212             52289       4399897
997          6431              567     109068             -4942       3417970
998         10543             1153     547446            109111       3926918
999         13788             1205     178553             59432       2049420

     Partnered  Mature     Language
0         True   False      English
1         True   False      English
2         True    True  Non-English
3         True   False      English
4         True   False      English
..         ...     ...          ...
995       True   False  Non-English
996       True   False  Non-English
997       True   False  Non-English
998       True   False      English
999       True   False  Non-English

[1000 rows x 11 columns]>
```

The lack of missing values means we can proceed

In [107]:

```
lgdata
```

Out[107]:

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	Views gained	Partnered	Mature	Lang
0	xQcOW	6196161750	215250	222720	27716	3246298	1734810	93036735	True	False	E

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	Views gained	Partnered	Mature	Lang
1	summit1g	6091677300	211845	310998	25610	5310163	1370184	89705964	True	False	E
2	Gaules	5644590915	515280	387315	10976	1767635	1023779	102611607	True	True	E
3	ESL_CSGO	3970318140	517740	300575	7714	3944850	703986	106546942	True	False	E
4	Tfue	3671000070	123660	285644	29602	8938903	2068424	78998587	True	False	E
...	...	...	...	...	...	...	...	...	...	...	
995	LITkillah	122524635	13560	21359	9104	601927	562691	2162107	True	False	E
996	빅헤드 (bighead033)	122523705	153000	3940	793	213212	52289	4399897	True	False	E
997	마스카 (newmasca)	122452320	217410	6431	567	109068	-4942	3417970	True	False	E
998	AndyMilonakis	122311065	104745	10543	1153	547446	109111	3926918	True	False	E
999	Remx	122192850	99180	13788	1205	178553	59432	2049420	True	False	E

1000 rows × 11 columns

In [108]:

```
lgdata = lgdata.drop(['Channel','Watch time(Minutes)','Stream time(minutes)','Followers
gained','Views gained','Followers gained','Views gained'], axis = 1)
```

In [109]:

```
lgdata
```

Out[109]:

	Peak viewers	Average viewers	Followers	Partnered	Mature	Language
0	222720	27716	3246298	True	False	English
1	310998	25610	5310163	True	False	English
2	387315	10976	1767635	True	True	Non-English
3	300575	7714	3944850	True	False	English
4	285644	29602	8938903	True	False	English
...	...	...	...	...	...	...
995	21359	9104	601927	True	False	Non-English
996	3940	793	213212	True	False	Non-English
997	6431	567	109068	True	False	Non-English
998	10543	1153	547446	True	False	English
999	13788	1205	178553	True	False	Non-English

1000 rows × 6 columns

In [110]:

```
lgdata.head().T
```

Out[110]:

	0	1	2	3	4
Peak viewers	222720	310998	387315	300575	285644
Average viewers	27716	25610	10976	7714	29602
Followers	3246298	5310163	1767635	3944850	8938903

	0	1	2	3	4
Partnered	True	True	True	True	True
Mature	False	False	True	False	False
Language	English	English	Non-English	English	English

Convert values to binary data

In [111]:

```
lgdata['Mature'] = lgdata['Mature'].replace([True], 1)
lgdata['Mature'] = lgdata['Mature'].replace([False], 0)
```

A 1 represents a True and a 0 represents a False

In [112]:

```
lgdata['Partnered'] = lgdata['Partnered'].replace([True], 1)
lgdata['Partnered'] = lgdata['Partnered'].replace([False], 0)
```

A 1 represents a True and a 0 represents a False

In [113]:

```
lgdata['Language'] = lgdata['Language'].replace(['English'], 1)
lgdata['Language'] = lgdata['Language'].replace(['Non-English'], 0)
```

A 1 represents an English-speaking streamer and 0 a non-English speaking streamer

In [114]:

```
lgdata['Followers'].where(lgdata['Followers'] > 1000000, 0, inplace=True)
lgdata['Followers'].where(lgdata['Followers'] <= 1000000, 1, inplace=True)
```

A 1 represents values equal to or above 1,000,000 and 0 represents values lower than 1,000,000

In [115]:

```
lgdata['Peak viewers'].where(lgdata['Peak viewers'] > 100000, 0, inplace=True)
lgdata['Peak viewers'].where(lgdata['Peak viewers'] <= 100000, 1, inplace=True)
```

A 1 represents values equal to or above 100,000 and 0 represents values lower than 100,000

In [116]:

```
lgdata['Average viewers'].where(lgdata['Average viewers'] > 10000, 0, inplace=True)
lgdata['Average viewers'].where(lgdata['Average viewers'] <= 10000, 1, inplace=True)
```

A 1 represents values equal to or above 10,000 and 0 represents values lower than 10,000

In [117]:

```
lgdata
```

Out[117]:

	Peak viewers	Average viewers	Followers	Partnered	Mature	Language
0	1	1	1	1	0	1
1	1	1	1	1	0	1
2	1	1	1	1	1	0
3	1	0	1	1	0	1
4	1	1	1	1	0	1

...	Peak viewers	Average viewers	Followers	Partnered	Mature	Language
995	0	0	0	1	0	0
996	0	0	0	1	0	0
997	0	0	0	1	0	0
998	0	0	0	1	0	1
999	0	0	0	1	0	0

1000 rows × 6 columns

In [118]:

```
lgdata.head().T
```

Out[118]:

	0	1	2	3	4
Peak viewers	1	1	1	1	1
Average viewers	1	1	1	0	1
Followers	1	1	1	1	1
Partnered	1	1	1	1	1
Mature	0	0	1	0	0
Language	1	1	0	1	1

In [119]:

```
lgdata['Followers'].unique
```

Out[119]:

```
<bound method Series.unique of 0       1
1       1
2       1
3       1
4       1
       ..
995     0
996     0
997     0
998     0
999     0
Name: Followers, Length: 1000, dtype: int64>
```

In [120]:

```
lgdata.isna().sum()
```

Out[120]:

```
Peak viewers       0
Average viewers    0
Followers          0
Partnered          0
Mature             0
Language           0
dtype: int64
```

Split the Dataset in accordance with the 80:20 rule

In [121]:

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

In [122]:

```
X=lgdata.iloc[:,0:30].values.astype(int)
```

In [123]:

```
X
```

Out[123]:

```
array([[1, 1, 1, 1, 0, 1],
       [1, 1, 1, 1, 0, 1],
       [1, 1, 1, 1, 1, 0],
       ...,
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 1],
       [0, 0, 0, 1, 0, 0]])
```

In [124]:

```
y=lgdata.iloc[:,5].values.astype(int)
```

In [125]:

```
y
```

Out[125]:

```
array([1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 0, 1, 0])
```

In [126]:

```
y.shape
```

Out[126]:

```
(1000,)
```

In [127]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=np.r
andom.seed(7))
```

In [128]:

```
X_train.shape
```

Out[128]:

```
(800, 6)
```

In [129]:

```
X_test.shape
```

Out[129]:

```
(200, 6)
```

Creating a LOgistic Regression Model

In [130]:

```
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, classificati
on_report
from sklearn.linear_model import LogisticRegression
import wandb
import time
```

Creating a reusable utility function that can be beneficial in the future

In [131]:

```
def train_eval_pipeline(model, train_data, test_data, name):
    #initialize wandb
    wandb.init(project = 'Machine Learning for Detecting Twitch Channel Language', name=n
ame)
    #assign the data
    (X_train, y_train)=train_data
    (X_test, y_test)=test_data

    #Train the model
    start=time.time()
    model.fit(X_train, y_train)
    end=time.time()-start
    prediction=model.predict(X_test)

    wandb.log({"accuracy":accuracy_score(y_test, prediction)*100, "precision":precision_
recall_fscore_support(y_test, prediction, average='macro')[0], "recall":precision_recall
_fscore_support(y_test, prediction, average='macro')[1], "training time": end})
    print("Accuracy Score of the Logistic Regression Classifier with default hyperparamet
er values {0:.2f}%".format(accuracy_score(y_test, prediction)*100.))
    print("\n")
    print("---Classification report of the Logistic Regression Classifier with default pa
rameter values---")
```

```
    print("\n")
    print(classification_report(y_test, prediction, target_names=["English", "Non-Englis
h"]))
```

In [132]:

```
logreg = LogisticRegression()
```

In [133]:

```
logreg
```

Out[133]:

```
LogisticRegression()
```

In [154]:

```
train_eval_pipeline(logreg, (X_train, y_train), (X_test, y_test), "Logistic_Regression_T
witch_Channel_Language")
```

Finishing last run (ID:2wby6h2q) before initializing another...

Waiting for W&B process to finish, PID 8604
Program ended successfully.

Find user logs for this run at: `c:\Users\keith\Desktop\Fall 2021\APT 3025 Machine Learning\wandb\run-20211009_234237-2wby6h2q\logs\debug.log`

Find internal logs for this run at: `c:\Users\keith\Desktop\Fall 2021\APT 3025 Machine Learning\wandb\run-20211009_234237-2wby6h2q\logs\debug-internal.log`

Run summary:

accuracy	**100.0**
precision	**1.0**
recall	**1.0**
training time	**0.003**
_runtime	**3**
_timestamp	**1633812165**
_step	**0**

Run history:

accuracy	▁
precision	▁
recall	▁
training time	▁
_runtime	▁
_timestamp	▁
_step	▁

Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)

Synced Logistic_Regression_Twitch_Channel_Language:
https://wandb.ai/flick/Machine%20Learning%20for%20Detecting%20Twitch%20Channel%20Language/runs/2wb

...Successfully finished last run (ID:2wby6h2q). Initializing new run:

```
wandb: wandb version 0.12.4 is available!  To upgrade, please run:
wandb:  $ pip install wandb --upgrade
```

Tracking run with wandb version 0.12.1
Syncing run Logistic_Regression_Twitch_Channel_Language to Weights & Biases (Documentation).
Project page:
https://wandb.ai/flick/Machine%20Learning%20for%20Detecting%20Twitch%20Channel%20Language
Run page:
https://wandb.ai/flick/Machine%20Learning%20for%20Detecting%20Twitch%20Channel%20Language/runs/51ss
Run data is saved locally in `c:\Users\keith\Desktop\Fall 2021\APT 3025 Machine Learning\wandb\run-20211009_234247-51ss924u`

```
Accuracy Score of the Logistic Regression Classifier with default hyperparameter values 1
00.00%


---Classification report of the Logistic Regression Classifier with default parameter val
ues---


              precision    recall  f1-score   support

     English       1.00      1.00      1.00       109
 Non-English       1.00      1.00      1.00        91

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

Can we improve the model?

A good way to start approaching is to tune the hyperparameters of the model. We need to define the grid of the values of the hyperparameters that we need to tune. We will use the random searchfor hyperparameter tuning.

Import GridSearchCV

In [135]:

```
from sklearn.model_selection import RandomizedSearchCV
```

Define the grid of values

In [136]:

```
penalty = ["l1", "l2"]
```

In [137]:

```
C = [0.8, 0.9, 1.0]
```

In [138]:

```
tol = [0.01, 0.001, 0.0001]
```

In [139]:

```
max_iter = [100, 150, 200, 250]
```

Create a dictionary where tol and max_iter are keys and lists of their values

are the corresponding values

In [140]:

```
param_grid = dict(penalty = penalty, C = C, tol = tol, max_iter = max_iter)
```

Now that we have the grid, we look for a set of hyperparameter values.

We instantiate RandomizedSearchCV with the search paramaters

In [141]:

```
random_model = RandomizedSearchCV(estimator=logreg, param_distributions=param_grid, cv=5
)
```

Fit the model to the data

In [142]:

```
random_model_results= random_model.fit(X_train, y_train)
```

Summarize the results

In [143]:

```
best_score, best_params = random_model_results.best_score_,random_model_results.best_para
ms_
```

In [144]:

```
print('Best score: %.2f using %s'%(best_score*100., best_params))
```

```
Best score: 100.00 using {'tol': 0.01, 'penalty': 'l2', 'max_iter': 150, 'C': 0.8}
```

Log the results of the hyperparameter in wandb

In [145]:

```
config = wandb.config
```

In [146]:

```
config.tol = 0.01
```

In [147]:

```
config.penalty = "l2"
```

In [148]:

```
config.C = 1.0
```

Train the model

In [149]:

```
logreg = LogisticRegression(tol=config.tol, penalty = config.penalty, max_iter = 250, C
= config.C)
```

In [150]:

```
logreg
```

Out[150]:

```
LogisticRegression(max_iter=250, tol=0.01)
```

In [157]:

```
train_eval_pipeline(logreg, (X_train,y_train), (X_test,y_test), "Logistic-Regression-Ran
dom-Search")
```

Finishing last run (ID:1c2qew54) before initializing another...

Waiting for W&B process to finish, PID 23340
Program ended successfully.

Find user logs for this run at: `c:\Users\keith\Desktop\Fall 2021\APT 3025 Machine Learning\wandb\run-20211009_234314-1c2qew54\logs\debug.log`

Find internal logs for this run at: `c:\Users\keith\Desktop\Fall 2021\APT 3025 Machine Learning\wandb\run-20211009_234314-1c2qew54\logs\debug-internal.log`

Run summary:

accuracy	**100.0**
precision	**1.0**
recall	**1.0**
training time	**0.003**
_runtime	**2**
_timestamp	**1633812202**
_step	**0**

Run history:

accuracy	**▁**
precision	**▁**
recall	**▁**
training time	**▁**
_runtime	**▁**
_timestamp	**▁**
_step	**▁**

Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)

Synced Logistic-Regression-Random-Search:
https://wandb.ai/flick/Machine%20Learning%20for%20Detecting%20Twitch%20Channel%20Language/runs/1c2q

...Successfully finished last run (ID:1c2qew54). Initializing new run:

```
wandb: wandb version 0.12.4 is available!  To upgrade, please run:
wandb:  $ pip install wandb --upgrade
```

Tracking run with wandb version 0.12.1
Syncing run Logistic-Regression-Random-Search to [Weights & Biases (Documentation)](https://wandb.ai).
Project page:
https://wandb.ai/flick/Machine%20Learning%20for%20Detecting%20Twitch%20Channel%20Language
Run page:
https://wandb.ai/flick/Machine%20Learning%20for%20Detecting%20Twitch%20Channel%20Language/runs/a8nb
Run data is saved locally in `c:\Users\keith\Desktop\Fall 2021\APT 3025 Machine Learning\wandb\run-20211009_234324-a8nblnll`

```
Accuracy Score of the Logistic Regression Classifier with default hyperparameter values 1
00.00%


---Classification report of the Logistic Regression Classifier with default parameter val
ues---


              precision    recall  f1-score   support

     English       1.00      1.00      1.00       109
 Non-English       1.00      1.00      1.00        91

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```