In [168]:

```
import pandas as pd
import math
import csv
import os
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [169]:

```
dfData = pd.read_csv("/home/archit/Desktop/ad vs organic/cleaned_subset.csv", encod
dfData.shape
```

Out[169]:

```
(10682, 13)
```

In [170]:

```
dfPrev = pd.read_csv("/home/archit/Desktop/ad vs organic/prev_vid_stat.csv")
dfPrev.columns = ['Id','PrevCommentCount', 'PrevDislikeCount', 'PrevLikeCount',
       'PrevPublishedAt', 'PrevViewCount', 'PrevTitle', 'PublishedYear',
       'ChannelAge']
dfPrev.to_csv("/home/archit/Desktop/ad vs organic/prev_vid_stat.csv", index=False)
dfPrev.shape
```

Out[170]:

```
(1670, 9)
```

In [171]:

```
dfLang = pd.read_csv("/home/archit/Desktop/ad vs organic/data_and_language.csv", en
dfLang.head()
```

Out[171]:

	Id	Lang
0	AUzyaHo0QQc	en
1	1Zgtdb7jp60	en
2	1Sfii7rnkJQ	en
3	UUwSKJjx9Go	en
4	Yl3NGvna2KA	en

In [172]:

```
dfChannel = pd.read_csv("/home/archit/Desktop/ad vs organic/channelStats.csv", enco
dfChannel.head()
```

Out[172]:

	Channel Id	publishedAt	subscriberCount	channelVideoCount	chan
0	UCUlTFib0pkPDGBYh7FQfo0A	2011-03-21T19:58:31.000Z	13	16	
1	UCvqHrRPqBw0D9B0wCNVwu8w	2012-01-07T22:35:00.000Z	3682	308	
2	UCatjfgWbdCUxNNAso8z9Usg	2006-10-06T22:31:17.000Z	96	50	
3	UCQa2_4V_9xtLefQGiPXqgNw	2006-10-12T09:10:36.000Z	29	4	
4	UCvzrgT1n8Im2bPogecOOu7A	2006-11-14T23:23:59.000Z	25	118	

In [173]:

```
dfData = dfData.merge(dfLang, on = 'Id', how = 'left')
dfData = dfData.merge(dfPrev, on = 'Id', how = 'left')
dfData = dfData.merge(dfChannel, on = 'Channel Id', how = 'left')
dfData.head()
```

Out[173]:

	Id	Title	Description	LikeCount	DislikeCount
0	AUzyaHo0QQc	b'300 pushups a day for 20 days!! - Results!!'	b'**NEW** (2016) Abs Workout for 30 Days | htt...	40408	10312
1	1Zgtdb7jp60	b'John Cena - gym'	b"Follow John Cena on twitter: http://www.twit...	37867	1486
2	1Sfii7rnkJQ	b'Bodybuilding Motivation - No Time To Waste'	b"Follow me:\nhttp://instagram.com/shaqx.bb\nh...	17688	1291
3	UUwSKJjx9Go	b'Most Powerful Home Chest Workout Ever : Buil...	b'http://www.6weeksixpack.com This is one of t...	46293	2589
4	YI3NGvna2KA	b'Greg Plitt Best of The Best Workout Video Pr...	b'SIGN UP TODAY - http://bit.ly/jointheranks\r...	24784	1134

5 rows × 26 columns

In [174]:

```
indices = list(dfPrev['Id'])
len(indices)
```

Out[174]:

```
1670
```

In [175]:

```
for index in range(0,len(indices)):
    #with open('/home/archit/Desktop/ad vs organic/prev_vid_stat.csv') as f:
    dfData.loc[dfData['Id'] == indices[index],].to_csv('/home/archit/Desktop/ad vs
```

In [176]:

```
dfData = pd.read_csv('/home/archit/Desktop/ad vs organic/cleaned_data4.csv')
dfData.columns = ['Id', 'Title', 'Description', 'LikeCount', 'DislikeCount', 'ViewC
       'FavoriteCount', 'CommentCount', 'PublishedAt', 'Channel Id',
       'Channel Title', 'Tags', 'Thumbnail Default', 'Lang',
       'PrevCommentCount', 'PrevDislikeCount', 'PrevLikeCount',
       'PrevPublishedAt', 'PrevViewCount', 'PrevTitle', 'PublishedYear',
       'ChannelAge', 'publishedAt', 'subscriberCount',
       'channelVideoCount', 'channelViewCount']
dfData.drop_duplicates(inplace = True)
```

In [177]:

```
dfData.columns
```

Out[177]:

```
Index(['Id', 'Title', 'Description', 'LikeCount', 'DislikeCount', 'Vie
wCount',
       'FavoriteCount', 'CommentCount', 'PublishedAt', 'Channel Id',
       'Channel Title', 'Tags', 'Thumbnail Default', 'Lang',
       'PrevCommentCount', 'PrevDislikeCount', 'PrevLikeCount',
       'PrevPublishedAt', 'PrevViewCount', 'PrevTitle', 'PublishedYea
r',
       'ChannelAge', 'publishedAt', 'subscriberCount', 'channelVideoCo
unt',
       'channelViewCount'],
      dtype='object')
```

In [178]:

```
dfData.head()
```

Out[178]:

	Id	Title	Description	LikeCount	DislikeCount
0	EBVTMSKY-Cw	b'BODYBUILDING MOTIVATION - How Bad Do You Wan...	b'Subscribe and Stay Tuned! Visit my channel a...	20440	1465
1	VnNH6OMqT9E	b'Jeff Seid Transformation 17 years old'	b"How ya doin' ;)\nwww.jeffseid.com\n\n\nLink ...	8807	2094
2	isb4txkVPrk	b'Bodybuilding Motivation - Collapse (MPW)'	b'Like/fav/sub! Visit http://www.cutandjacked....	4686	353
3	o-IVVhPrZ0A	b'BODYBUILDING TILL IM DEAD'	b'http://www.professionalmuscle.com - Chat Liv...	15630	496
4	L4S7sYup_Rw	b'Stomach Exercises For Sexy Abs'	b'http://www.2losebellyfat.com/ - Visit for mo...	5912	223

5 rows × 26 columns

In [179]:

```
dfData = dfData[dfData['channelVideoCount'] < 2000]
dfData = dfData[dfData['subscriberCount'] > 0]
dfData = dfData[dfData['Lang'] == 'en']
```

In [180]:

```
dfData.shape
```

Out[180]:

```
(1656, 26)
```

In [181]:

```
type(dfData['Title'])
```

Out[181]:

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

In [182]:

```
# Conver to lower case
dfData['Title'] = dfData['Title'].str.lower()


# How To
dfData['HowTo'] = (dfData['Title'].str.contains('how')  |
                       dfData['Tags'].str.contains('how'))

# Motivational
dfData['Motivation'] = (dfData['Title'].str.contains('motivation')  |
                       dfData['Tags'].str.contains('motivation')
                      )


# Transformation
dfData['Transform'] = (dfData['Title'].str.contains('transform')  |
                       dfData['Tags'].str.contains('transform')
                      )

# Abs Workout
dfData['Abs Video'] = (dfData['Title'].str.contains('abs')  |
                       dfData['Title'].str.contains('six') |
                       dfData['Title'].str.contains(' 6') |
                       dfData['Title'].str.contains('abdomen') |
                       dfData['Tags'].str.contains('abs')  |
                       dfData['Tags'].str.contains('six') |
                       dfData['Tags'].str.contains(' 6') |
                       dfData['Tags'].str.contains('abdomen')
                      )

# Chest Workout
dfData['Chest Video'] = (dfData['Title'].str.contains('chest')  |
                         dfData['Title'].str.contains('pushup') |
                         dfData['Title'].str.contains('bench') |
                         dfData['Title'].str.contains('bench') |
                         dfData['Title'].str.contains('push up') |
                         dfData['Title'].str.contains('dumbell press') |
                         dfData['Tags'].str.contains('chest')  |
                         dfData['Tags'].str.contains('pushup') |
                         dfData['Tags'].str.contains('bench') |
                         dfData['Tags'].str.contains('bench') |
                         dfData['Tags'].str.contains('push up') |
                         dfData['Tags'].str.contains('dumbell press'))

# Back Workouts
dfData['Back Video'] = (dfData['Title'].str.contains('back') |
                        dfData['Title'].str.contains('pull up') |
                        dfData['Title'].str.contains('chin up')  |
                        dfData['Title'].str.contains('deadlift') |
                        dfData['Tags'].str.contains('back') |
                        dfData['Tags'].str.contains('pull up') |
                        dfData['Tags'].str.contains('chin up')  |
                        dfData['Tags'].str.contains('deadlift'))

# Leg Workouts
dfData['Legs Video'] = (dfData['Title'].str.contains('leg') |
                        dfData['Title'].str.contains('squat') |
                        dfData['Title'].str.contains('butt') |
                        dfData['Title'].str.contains('quad') |
                        dfData['Title'].str.contains('calve') |
                       dfData['Tags'].str.contains('leg') |
                        dfData['Tags'].str.contains('squat') |
                        dfData['Tags'].str.contains('butt') |
                        dfData['Tags'].str.contains('quad') |
                        dfData['Tags'].str.contains('calve') )

# Arm Workout
dfData['Arm Video'] = (dfData['Title'].str.contains('shoulder') |
                       dfData['Title'].str.contains('arm') |
                       dfData['Title'].str.contains('bicep')  |
                       dfData['Title'].str.contains('tricep') |
                       dfData['Title'].str.contains('delt') |
                       dfData['Tags'].str.contains('shoulder') |
                       dfData['Tags'].str.contains('arm') |
                       dfData['Tags'].str.contains('bicep')  |
                       dfData['Tags'].str.contains('tricep') |
                       dfData['Tags'].str.contains('delt'))
```

In [183]:

```
dfData.head()
```

Out[183]:

	Id	Title	Description	LikeCount	DislikeCount	Vie
0	EBVTMSKY-Cw	b'bodybuilding motivation - how bad do you wan...	b'Subscribe and Stay Tuned! Visit my channel a...	20440	1465	(
1	VnNH6OMqT9E	b'jeff seid transformation 17 years old'	b"How ya doin' ;)\nwww.jeffseid.com\n\n\nLink ...	8807	2094	:
2	isb4txkVPrk	b'bodybuilding motivation - collapse (mpw)'	b'Like/fav/sub! Visit http://www.cutandjacked....	4686	353	:
3	o-IVVhPrZ0A	b'bodybuilding till im dead'	b'http://www.professionalmuscle.com - Chat Liv...	15630	496	:
4	L4S7sYup_Rw	b'stomach exercises for sexy abs'	b'http://www.2losebellyfat.com/ - Visit for mo...	5912	223	:

5 rows × 34 columns

In [184]:

```
# order of preference in categorizing video in case there are multiple categories t
# abs < chest < back < legs < arm < motivaton < HowTo < Motivation < Transform
dfData.loc[dfData['Abs Video'] == True , 'Category'] = 'Abs'
dfData.loc[dfData['Chest Video'] == True , 'Category'] = 'Chest'
dfData.loc[dfData['Back Video'] == True, 'Category'] = 'Back'
dfData.loc[dfData['Legs Video'] == True, 'Category'] = 'Legs'
dfData.loc[dfData['Arm Video'] == True, 'Category'] = 'Arms'
#dfData.loc[dfData['Motivation'] == True, 'Category'] = 'Motivation'
dfData.loc[dfData['HowTo'] == True, 'Category'] = 'HowTo'
dfData.loc[dfData['Motivation'] == True, 'Category'] = 'Motivation'
dfData.loc[dfData['Transform'] == True, 'Category'] = 'Transform'
dfData.shape
```

Out[184]:

```
(1656, 35)
```

In [185]:

```
otherVid = dfData[dfData['Category'].isnull()]
otherVid.shape[0]
```

Out[185]:

```
886
```

In [186]:

```
dfData = dfData[dfData['Category'].notnull()]
dfData.shape
```

Out[186]:

```
(770, 35)
```

In [187]:

```
absVid = dfData[dfData['Category'] == 'Abs']
chestVid = dfData[dfData['Category'] == 'Chest']
backVid = dfData[dfData['Category'] == 'Back']
legsVid = dfData[dfData['Category'] == 'Legs']
armsVid = dfData[dfData['Category'] == 'Arms']
howToVid = dfData[dfData['Category'] == 'HowTo']
motivationVid = dfData[dfData['Category'] == 'Motivation']
transformVid = dfData[dfData['Category'] == 'Transform']


print("Number of Abs Related Videos: " + str(absVid.shape[0]))
print("Number of Chest Related Videos: " + str(chestVid.shape[0]))
print("Number of Back Related Videos: " + str(backVid.shape[0]))
print("Number of Leg Related Videos: " + str(legsVid.shape[0]))
print("Number of Arm Related Videos: " + str(armsVid.shape[0]))
```

```
Number of Abs Related Videos: 50
Number of Chest Related Videos: 60
Number of Back Related Videos: 80
Number of Leg Related Videos: 123
Number of Arm Related Videos: 140
```

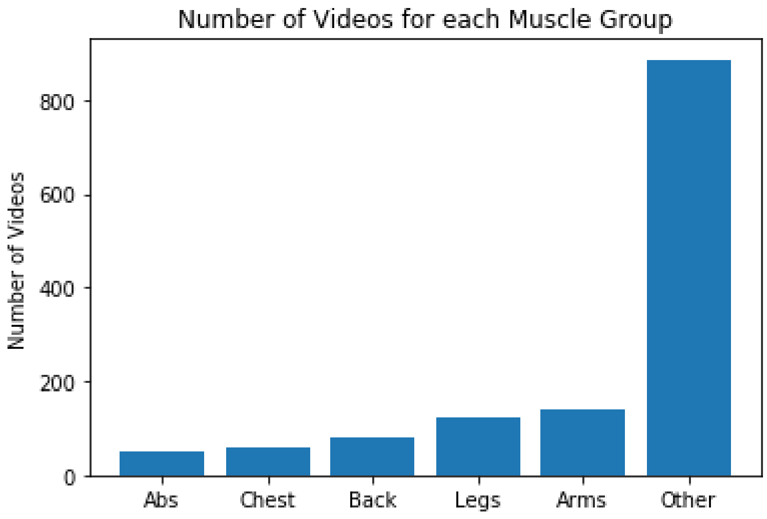
Different categories of fitnesss videos in database

In [188]:

```
yCols = ['Abs', 'Chest', 'Back', 'Legs', 'Arms', 'Other']
xCols = [absVid.shape[0], chestVid.shape[0], backVid.shape[0], legsVid.shape[0], arm
y = np.arange(len(yCols))
plt.bar(y, xCols,align = 'center')
plt.xticks(y,yCols)
plt.ylabel("Number of Videos")
plt.title("Number of Videos for each Muscle Group")
```

Out[188]:

```
Text(0.5,1,'Number of Videos for each Muscle Group')
```

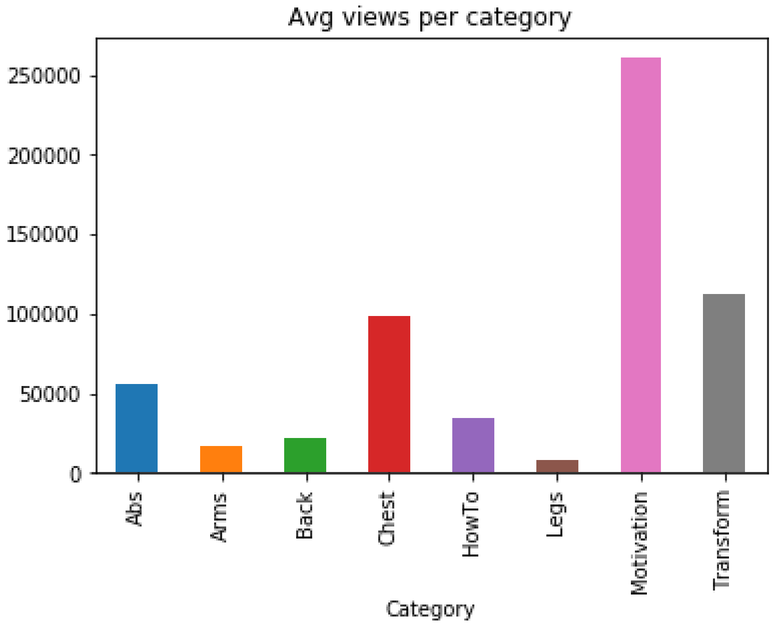


Understanding view distribution

In [189]:

```
dfData.groupby('Category').ViewCount.mean().plot(kind = 'bar', title = 'Avg views p
print(dfData.groupby('Category').ViewCount.mean().sort_values(ascending = False))
```

```
Category
Motivation    260460.650000
Transform     112217.136364
Chest          98683.400000
Abs            55900.980000
HowTo          34247.973856
Back           21723.912500
Arms           17165.078571
Legs            8080.227642
Name: ViewCount, dtype: float64
```



In [190]:

```
dfData.columns
```

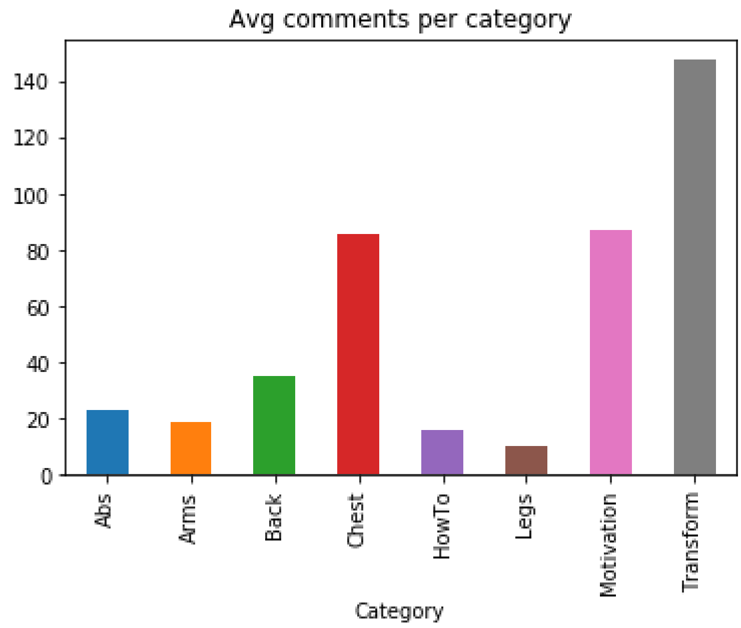
Out[190]:

```
Index(['Id', 'Title', 'Description', 'LikeCount', 'DislikeCount', 'Vie
wCount',
       'FavoriteCount', 'CommentCount', 'PublishedAt', 'Channel Id',
       'Channel Title', 'Tags', 'Thumbnail Default', 'Lang',
       'PrevCommentCount', 'PrevDislikeCount', 'PrevLikeCount',
       'PrevPublishedAt', 'PrevViewCount', 'PrevTitle', 'PublishedYea
r',
       'ChannelAge', 'publishedAt', 'subscriberCount', 'channelVideoCo
unt',
       'channelViewCount', 'HowTo', 'Motivation', 'Transform', 'Abs Vi
deo',
       'Chest Video', 'Back Video', 'Legs Video', 'Arm Video', 'Catego
ry'],
      dtype='object')
```

In [191]:

```
dfData.groupby('Category').CommentCount.mean().plot(kind = 'bar', title = 'Avg comm
print(dfData.groupby('Category').ViewCount.mean().sort_values(ascending = False))
```

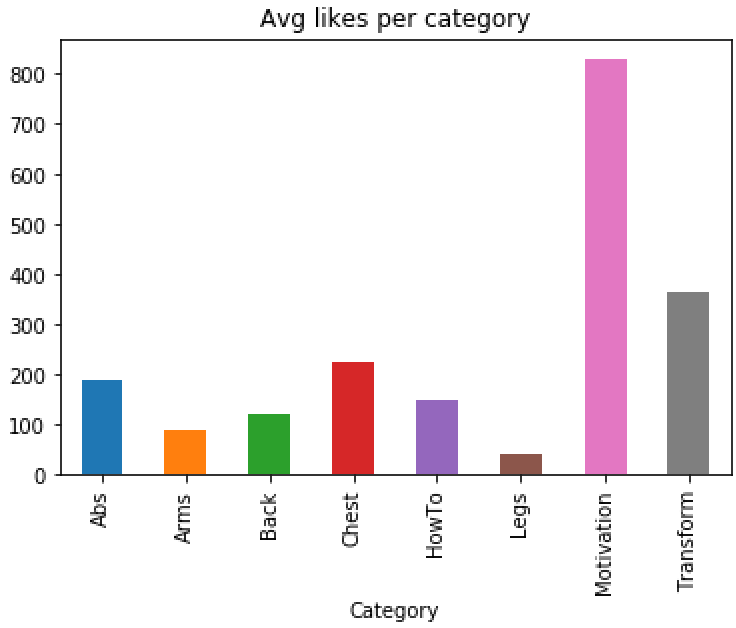
```
Category
Motivation    260460.650000
Transform     112217.136364
Chest          98683.400000
Abs            55900.980000
HowTo          34247.973856
Back           21723.912500
Arms           17165.078571
Legs            8080.227642
Name: ViewCount, dtype: float64
```



In [192]:

```
dfData.groupby('Category').LikeCount.mean().plot(kind = 'bar', title = 'Avg likes p
print(dfData.groupby('Category').LikeCount.mean().sort_values(ascending = False))
```

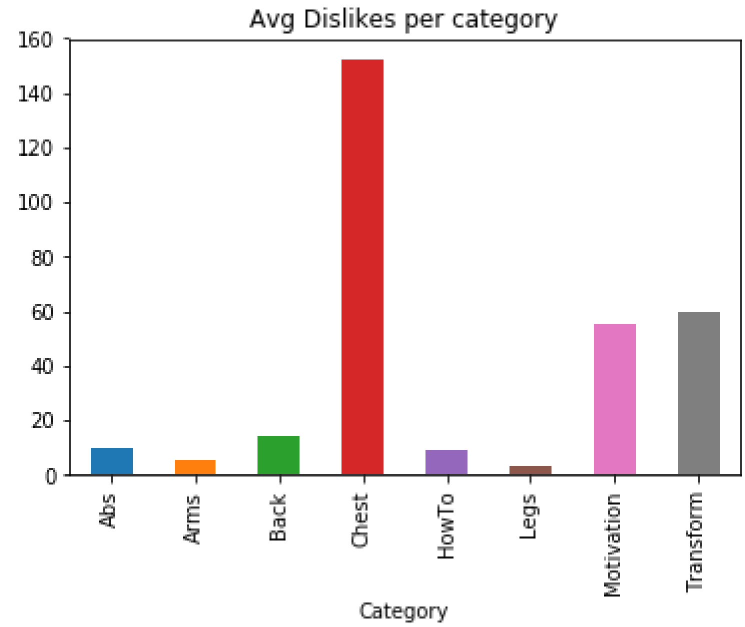
```
Category
Motivation    828.575000
Transform     364.863636
Chest         224.150000
Abs           189.080000
HowTo         149.921569
Back          119.637500
Arms           91.157143
Legs           42.536585
Name: LikeCount, dtype: float64
```



In [193]:

```
dfData.groupby('Category').DislikeCount.mean().plot(kind = 'bar', title = 'Avg Disl
print(dfData.groupby('Category').DislikeCount.mean().sort_values(ascending = False)
```

```
Category
Chest         152.100000
Transform      59.818182
Motivation     55.358333
Back           13.862500
Abs             9.420000
HowTo           9.287582
Arms            5.371429
Legs            3.333333
Name: DislikeCount, dtype: float64
```



Obseravtions

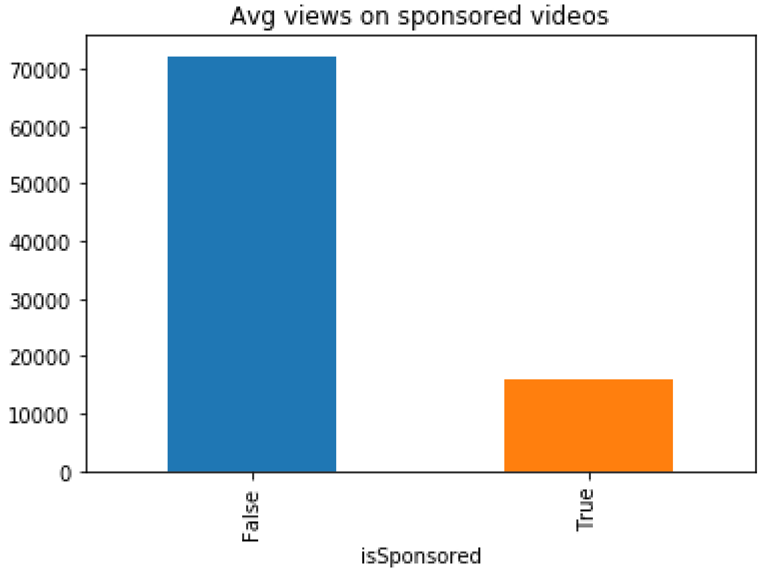
Give proper insight here later using venn diagram

Sponsorship/ discounted videos view distribution

In [194]:

```
dfData['isSponsored'] = dfData['Description'].str.contains("sponsored") | dfData['D
dfData.groupby('isSponsored').ViewCount.mean().plot(kind = 'bar', title = 'Avg view
print(dfData.groupby('isSponsored').ViewCount.mean())
```

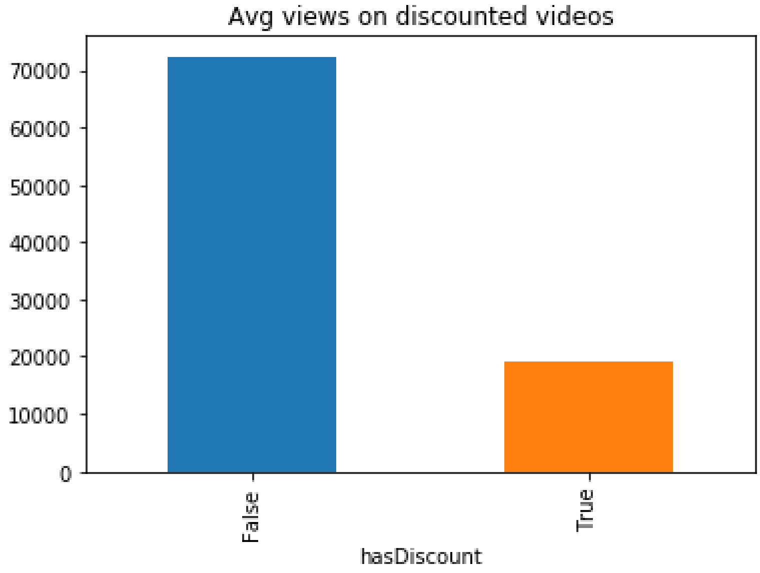
```
isSponsored
False    72310.131062
True     15869.571429
Name: ViewCount, dtype: float64
```



In [195]:

```
dfData['hasDiscount'] = dfData['Description'].str.contains('discount') | dfData['De
dfData.groupby('hasDiscount').ViewCount.mean().plot(kind = 'bar', title = 'Avg view
print(dfData.groupby('hasDiscount').ViewCount.mean())
```

```
hasDiscount
False    72421.400788
True     19003.444444
Name: ViewCount, dtype: float64
```



In [196]:

```
dfData['LikeDislikeRatio'] = dfData['LikeCount']/(dfData['LikeCount'] + dfData['Dis
```

In [197]:

```
dfData.shape
```

Out[197]:

```
(770, 38)
```

In [198]:

```
dfData.columns
```

Out[198]:

```
Index(['Id', 'Title', 'Description', 'LikeCount', 'DislikeCount', 'Vie
wCount',
       'FavoriteCount', 'CommentCount', 'PublishedAt', 'Channel Id',
       'Channel Title', 'Tags', 'Thumbnail Default', 'Lang',
       'PrevCommentCount', 'PrevDislikeCount', 'PrevLikeCount',
       'PrevPublishedAt', 'PrevViewCount', 'PrevTitle', 'PublishedYea
r',
       'ChannelAge', 'publishedAt', 'subscriberCount', 'channelVideoCo
unt',
       'channelViewCount', 'HowTo', 'Motivation', 'Transform', 'Abs Vi
deo',
       'Chest Video', 'Back Video', 'Legs Video', 'Arm Video', 'Catego
ry',
       'isSponsored', 'hasDiscount', 'LikeDislikeRatio'],
      dtype='object')
```

In [199]:

```
dfData):
op(['Category','FavoriteCount', 'PrevTitle', 'PrevPublishedAt'],axis =1)
h'] = dfData['Lang'].apply(lambda x: 1 if x=='en' else 0)
rop('Lang', axis=1)

= dfData['Abs Video'].apply(lambda x: 1 if x==True else 0)
] = dfData['Chest Video'].apply(lambda x: 1 if x==True else 0)
 = dfData['Back Video'].apply(lambda x: 1 if x==True else 0)
 = dfData['Legs Video'].apply(lambda x: 1 if x==True else 0)
 = dfData['Arm Video'].apply(lambda x: 1 if x==True else 0)
ed'] = dfData['isSponsored'].apply(lambda x: 1 if x==True else 0)
nt'] = dfData['hasDiscount'].apply(lambda x: 1 if x==True else 0)
Year'] = dfData['PublishedAt'].apply(lambda x: x[:4])
e'] =  dfData['publishedAt'].apply(lambda x: x[:4])
ewCount'] = np.log(dfData['channelViewCount'])
keRatio'] = dfData['LikeCount']/(dfData['DislikeCount'] + dfData['LikeCount'])
ntCount'] = dfData['PrevCommentCount'].fillna(0)
keCount'] = dfData['PrevDislikeCount'].fillna(0)
ount'] = dfData['PrevLikeCount'].fillna(0)
ount'] = dfData['PrevViewCount'].fillna(0)

keRatio'] = dfData['LikeDislikeRatio'].replace(np.inf, np.nan)
keRatio'] = dfData['LikeDislikeRatio'].fillna(0)
rop(['Title','Description','PublishedAt','publishedAt','Channel Id','Channel Title'



ures(dfData):
rop(['LikeCount','DislikeCount','CommentCount'],axis = 1)
```

In [200]:

```
df = feature_engineer(dfData)
df.columns
```

Out[200]:

```
Index(['Id', 'LikeCount', 'DislikeCount', 'ViewCount', 'CommentCount',
       'PrevCommentCount', 'PrevDislikeCount', 'PrevLikeCount',
       'PrevViewCount', 'PublishedYear', 'ChannelAge', 'subscriberCoun
t',
       'channelVideoCount', 'channelViewCount', 'HowTo', 'Motivation',
       'Transform', 'isSponsored', 'hasDiscount', 'LikeDislikeRatio',
       'is_english', 'is_Abs', 'is_Chest', 'is_Back', 'is_Legs', 'is_A
rms'],
      dtype='object')
```

In [201]:

```
df.columns
```

Out[201]:

```
Index(['Id', 'LikeCount', 'DislikeCount', 'ViewCount', 'CommentCount',
       'PrevCommentCount', 'PrevDislikeCount', 'PrevLikeCount',
       'PrevViewCount', 'PublishedYear', 'ChannelAge', 'subscriberCoun
t',
       'channelVideoCount', 'channelViewCount', 'HowTo', 'Motivation',
       'Transform', 'isSponsored', 'hasDiscount', 'LikeDislikeRatio',
       'is_english', 'is_Abs', 'is_Chest', 'is_Back', 'is_Legs', 'is_A
rms'],
      dtype='object')
```

In [202]:

```
X, y = df.drop('ViewCount',axis = 1) , np.log(df['ViewCount'])
X.head()
```

Out[202]:

	Id	LikeCount	DislikeCount	CommentCount	PrevCommentCount	PrevDislikeCo
0	EBVTMSKY-Cw	20440	1465	2397	159.0	10
1	VnNH6OMqT9E	8807	2094	4598	0.0	
2	isb4txkVPrk	4686	353	901	43.0	
3	o-IVVhPrZ0A	15630	496	1928	1037.0	19
4	L4S7sYup_Rw	5912	223	456	0.0	

5 rows × 25 columns

In [203]:

```
X.columns
```

Out[203]:

```
Index(['Id', 'LikeCount', 'DislikeCount', 'CommentCount', 'PrevComment
Count',
       'PrevDislikeCount', 'PrevLikeCount', 'PrevViewCount', 'Publishe
dYear',
       'ChannelAge', 'subscriberCount', 'channelVideoCount',
       'channelViewCount', 'HowTo', 'Motivation', 'Transform', 'isSpon
sored',
       'hasDiscount', 'LikeDislikeRatio', 'is_english', 'is_Abs', 'is_
Chest',
       'is_Back', 'is_Legs', 'is_Arms'],
      dtype='object')
```

Modeling training data with GradientBoostedClassifier

linear regression does not work great

In [204]:

```
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.model_selection import train_test_split
#from xgboost import XGBClassifier, XGBRegressor
```

In [208]:

```
#train_X, test_X, train_y, test_y = train_test_split(X, y, train_size = 0.75, test_

X_id, X = X.Id, X.drop('Id', axis = 1)

reg = GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
             min_impurity_split=None, min_samples_leaf=1,
             min_samples_split=5, min_weight_fraction_leaf=0.0,
             n_estimators=100, presort='auto', random_state=None,
             subsample=0.5, verbose=0, warm_start=False)
reg.fit(X, y)
reg.score(X, y)
#pred = reg.predict(X)

#import pickle
#pickle.dump(reg, open("Gbr002.pickle.dat", "wb"))
```

Out[208]:

```
0.9406991654797717
```

In [209]:

```
pred = reg.predict(X)
```

In [210]:

```
# function exponentiates values in test_y and pred and
# than calulated rmse of actual views
size = y.shape[0]
test_yL = y.tolist()
predL = pred.tolist()
X_id = X_id.tolist()
exActView = []
exPredView = []
diff = []
mape = {}
sqErr = []
for i in range(0, size):
    exActView.append(math.exp(test_yL[i]))
    exPredView.append(math.exp(predL[i]))
    diff.append(exActView[i] - exPredView[i])
    sqErr.append(math.pow(diff[i], 2))
for i in range(0, len(diff)):
    try:
        mape[X_id[i]] = abs(diff[i]/exActView[i])
    except KeyError:
        pass
#Mape = sum(mape)/len(diff)
rmse = math.sqrt(sum(sqErr) /size)/(max(exActView) - min(exActView))
dfEval = pd.DataFrame({'Id':X_id, 'Actual' : exActView, 'Pred': exPredView, 'Diff':
dfEval.head()
```

Out[210]:

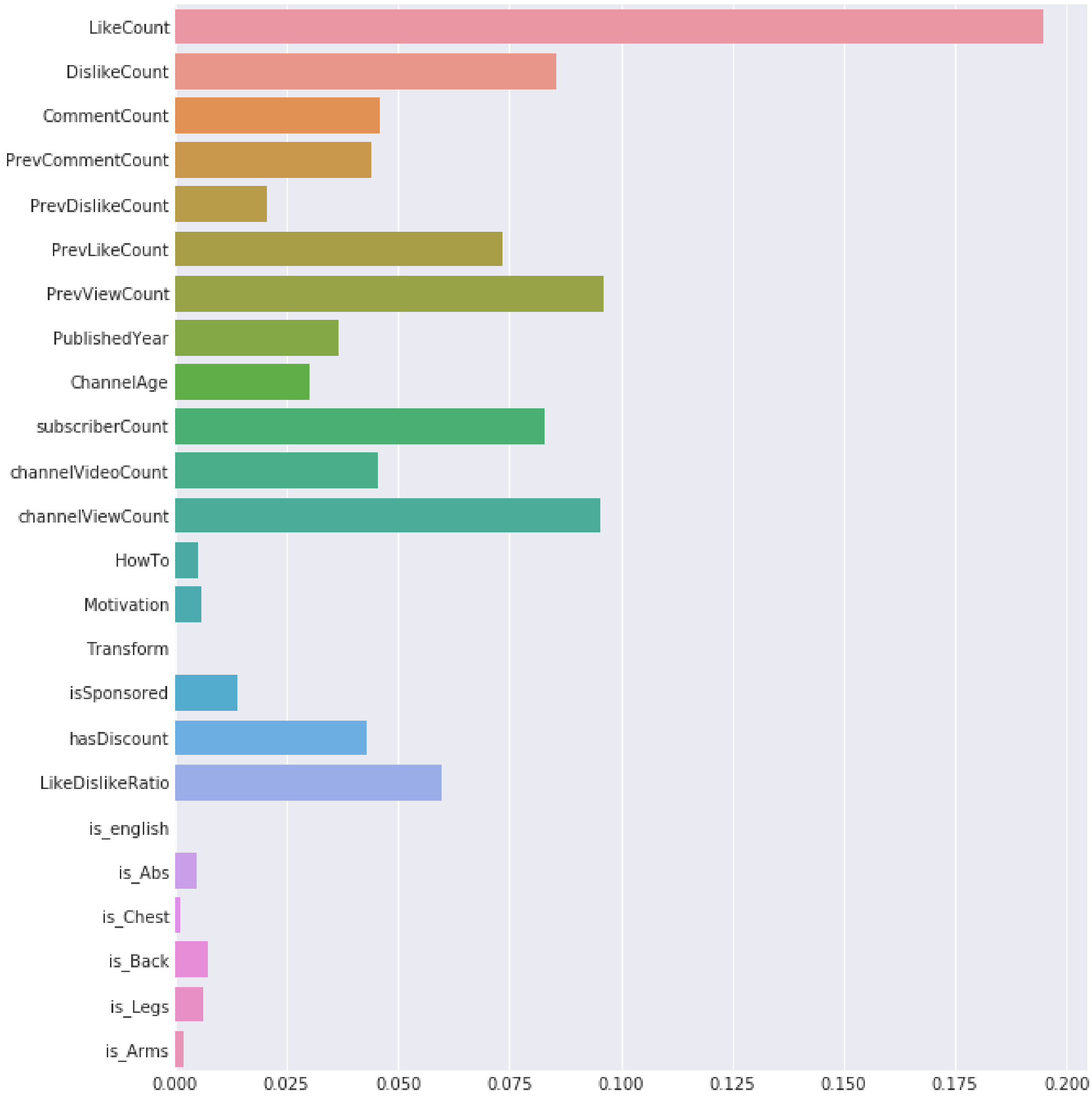
	Actual	Diff	Id	Pred
0	6799110.0	1.923742e+06	EBVTMSKY-Cw	4.875368e+06
1	3669026.0	3.904099e+05	VnNH6OMqT9E	3.278616e+06
2	3477038.0	9.692511e+05	isb4txkVPrk	2.507787e+06
3	3191046.0	-1.112718e+05	o-IVVhPrZ0A	3.302318e+06
4	2211906.0	3.104747e+05	L4S7sYup_Rw	1.901431e+06

In [212]:

```
#X = X.drop('Id', axis = 1)
sns.set_style('darkgrid')
plt.figure(figsize=(10,12))
sns.barplot(x=reg.feature_importances_, y=X.columns)
```

Out[212]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f10aeba6f60>
```



In [213]:

```
# using train test split
X, y = df.drop(['ViewCount'], axis = 1), np.log(df['ViewCount'])
train_X, test_X, train_y, test_y = train_test_split(X, y, train_size = 0.75, test_s

train_X_id, train_X = train_X.Id, train_X.drop('Id', axis = 1)
test_X_id, test_X = test_X.Id, test_X.drop('Id', axis = 1)

reg = GradientBoostingRegressor()
reg.fit(train_X, train_y)
reg.score(test_X, test_y)
```

Out[213]:

```
0.7986361572854548
```

In [214]:

```
pred = reg.predict(test_X)
```

In [215]:

```
size = test_y.shape[0]
test_yL = test_y.tolist()
predL = pred.tolist()
exActView = []
exPredView = []
diff = []
sqErr = []
mape={}
for i in range(0, size):
    exActView.append(math.exp(test_yL[i]))
    exPredView.append(math.exp(predL[i]))
    diff.append(exActView[i] - exPredView[i])
    sqErr.append(math.pow(diff[i], 2))
rmse = math.sqrt(sum(sqErr) /size)/ (test_y.max() - test_y.min())
dfEval = pd.DataFrame({'Id':test_X_id, 'Actual' : exActView, 'Pred': exPredView, 'D
dfEval.head()
dfEval.to_csv('/home/archit/Desktop/ad vs organic/predictions_data.csv')
```

In [216]:

```
print("Rmse of the regressormodel is:" + str(rmse))
```

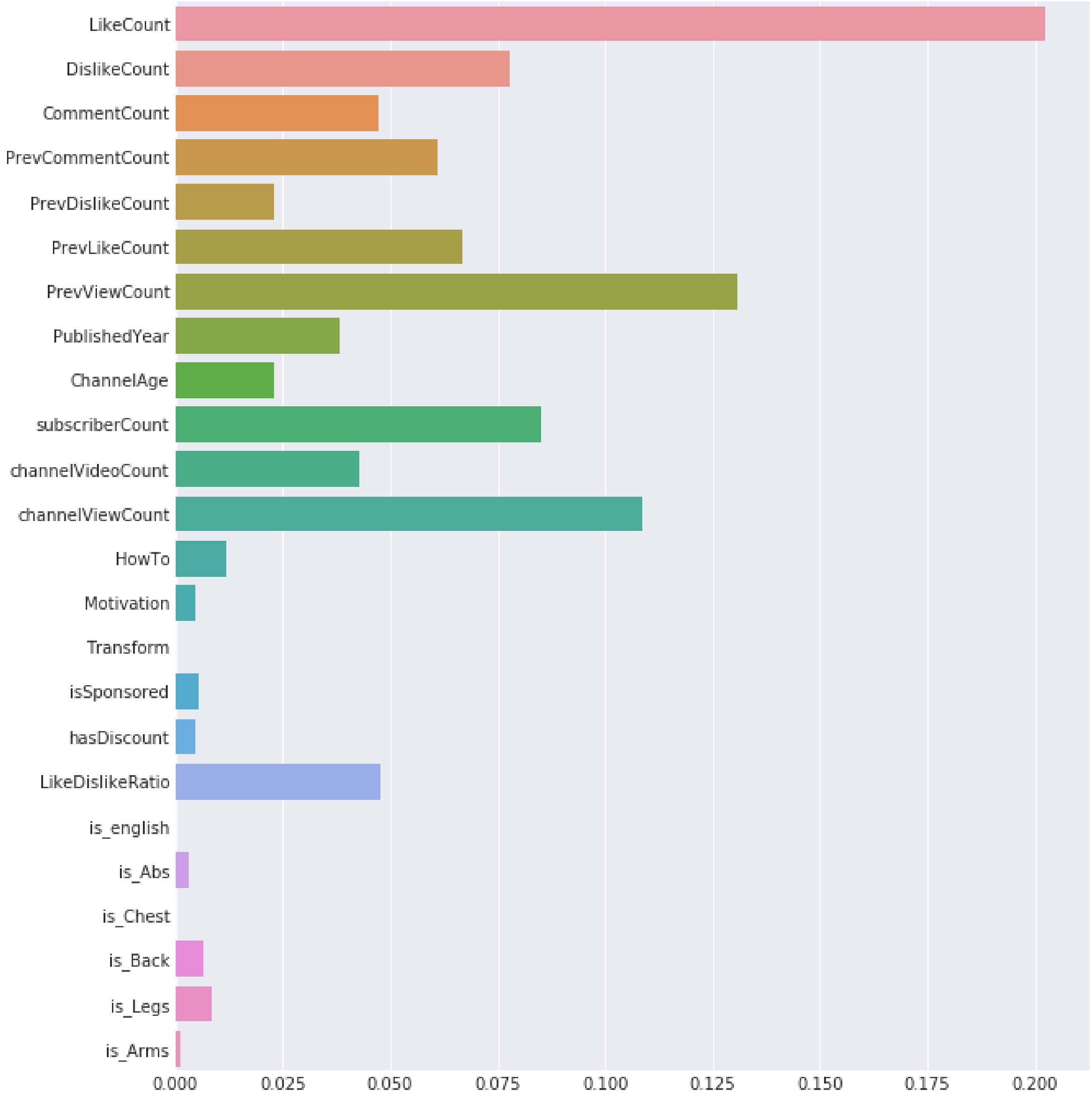
```
Rmse of the regressormodel is:17527.720394091688
```

In [218]:

```
X = X.drop('Id', axis = 1)
sns.set_style('darkgrid')
plt.figure(figsize=(10,12))
sns.barplot(x=reg.feature_importances_, y=X.columns)
```

Out[218]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f10af265e48>
```



In [219]:

```
df = drop_obvious_features(feature_engineer(dfData))
X, y = df.drop(['ViewCount', 'LikeDislikeRatio'], axis = 1), np.log(df['ViewCount']
train_X, test_X, train_y, test_y = train_test_split(X, y, train_size = 0.75, test_s

train_X_id, train_X = train_X.Id, train_X.drop('Id', axis = 1)
test_X_id, test_X = test_X.Id, test_X.drop('Id', axis = 1)

reg = GradientBoostingRegressor()
reg.fit(train_X, train_y)
reg.score(test_X, test_y)
```

Out[219]:

```
0.3920827275692862
```

In [220]:

```
pred = reg.predict(test_X)
```

In [221]:

```
# function exponentiates values in test_y and pred and
# than calulated rmse of actual views
size = test_y.shape[0]
test_yL = test_y.tolist()
predL = pred.tolist()
exActView = []
exPredView = []
diff = []
sqErr = []
for i in range(0, size):
    exActView.append(math.exp(test_yL[i]))
    exPredView.append(math.exp(predL[i]))
    diff.append(exActView[i] - exPredView[i])
    sqErr.append(math.pow(diff[i], 2))
rmse = math.sqrt(sum(sqErr) /size)/ (test_y.max() - test_y.min())
dfEval = pd.DataFrame({'Actual' : exActView, 'Pred': exPredView, 'Diff': diff})
dfEval.head()
```

Out[221]:

	Actual	Diff	Pred
0	1318.0	-2334.125662	3652.125662
1	1726.0	-9496.774108	11222.774108
2	9277.0	-287.219986	9564.219986
3	66202.0	-81105.936912	147307.936912
4	22523.0	12253.996110	10269.003890

In [222]:

```
print("Rmse of the regressormodel is:" + str(rmse))
```

```
Rmse of the regressormodel is:23244.5928813892
```

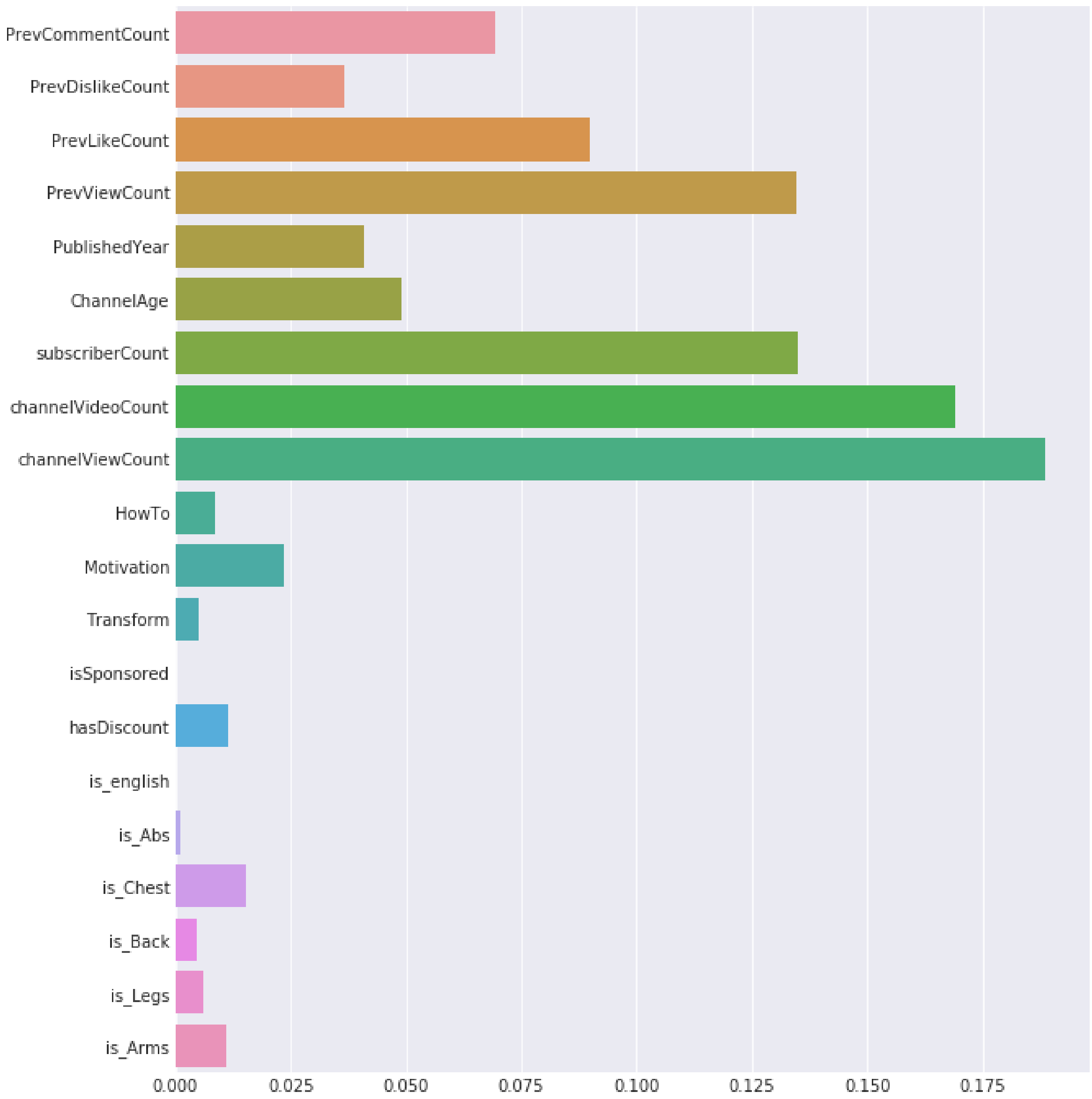
In [ ]:

In [223]:

```
X = X.drop('Id', axis = 1)
sns.set_style('darkgrid')
plt.figure(figsize=(10,12))
sns.barplot(x=reg.feature_importances_, y=X.columns)
```

Out[223]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f10bbd1cb00>
```



In [ ]: