In [17]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import os
import re
from matplotlib import pyplot as plt
import seaborn as sns
import glob
import re
from scipy.interpolate import interp1d
from tqdm import tqdm, tqdm_notebook
from scipy import spatial
from sklearn.linear_model import LogisticRegression
```

In [5]:
```
def calculate_pad(brightness, saturation):
    p = 0.69*brightness + 0.22*saturation
    a = -0.31*brightness + 0.6*saturation
    d = 0.76*brightness + 0.32*saturation
    return [p,a,d]

def calculate_pad_scene(scene):
    pads = []
    for img in scene:
        temp_b = mean_brightness(img)
        temp_s = mean_saturation(img)
        pads.append(calculate_pad(temp_b, temp_s))
    return np.mean([x[0] for x in pads]), np.mean([x[1] for x in pads]), np.mean

def calculate_blur(img):
    return cv2.Laplacian(img, cv2.CV_64F).var()

def calculate_blur_scene(scene):
    blurs = []
    for img in scene:
        blurs.append(calculate_blur(img))
    return np.mean(blurs)

def mean_brightness(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) #convert it to hsv
    return np.mean(hsv[:,:,2])

def mean_saturation(img):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) #convert it to hsv
    return np.mean(hsv[:,:,1])

def calculate_opticalFlow(img1, img2):
    prev = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    forward = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    mask = np.zeros_like(prev)
    mask[..., 1] = 255
    flow = cv2.calcOpticalFlowFarneback(prev, forward, flow=None, pyr_scale=0.5,
    magnitude, angle = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    return cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX)[0]

def calculate_opticalFlow_scene(scene):
    first = calculate_opticalFlow(scene[0], scene[1])
    second = calculate_opticalFlow(scene[1], scene[2])
    return np.mean([first, second])

def isjpg(filepath):
    return re.search(".jpg$", filepath)

def calculate_features(img_path):
    img = cv2.imread(img_path)
    b = mean_brightness(img)
    s = mean_saturation(img)
    result = calculate_pad(b, s)
    result.append(calculate_blur(img))
    return result
```

In [14]:
```
base = 'data\\outside\\training'
dirs = os.listdir(base)
labels = []
train_ps = []
train_as = []
train_ds = []
train_blurs = []
for emotion in dirs:
    emotion_path = os.path.join(base, emotion)
    for img in tqdm_notebook(os.listdir(emotion_path)):
        img_path = os.path.join(emotion_path, img)
        res = calculate_features(img_path)
        labels.append(emotion)
        train_ps.append(res[0])
        train_as.append(res[1])
        train_ds.append(res[2])
        train_blurs.append(res[3])
```

100% 77/77 [00:01<00:00, 60.60it/s]

100% 70/70 [00:01<00:00, 66.03it/s]

100% 115/115 [00:01<00:00, 67.23it/s]

100% 101/101 [00:01<00:00, 66.67it/s]

100% 166/166 [00:02<00:00, 66.55it/s]

100% 102/102 [00:01<00:00, 66.20it/s]

In [15]:
```
df = pd.DataFrame()
df['label'] = labels
df['valence'] = train_ps
df['arousal'] = train_as
df['dominance'] = train_ds
df['blur'] = train_blurs
```

In [24]:
```
X = df.drop(columns=['label'])
X = (X - X.min())/(X.max() - X.min())
Y = df.label
clf = LogisticRegression(random_state=0).fit(X, Y)
```

```
C:\Users\xinrui zhan\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.
py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
C:\Users\xinrui zhan\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.
py:469: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. S
pecify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

In [29]:
```
our_dataset = pd.read_csv('clean_df.csv')
our_X = our_dataset[['scene_avg_p', 'scene_avg_a', 'scene_avg_d', 'scene_avg_blu
our_X = (our_X - our_X.min())/(our_X.max() - our_X.min())
prd = clf.predict(our_X)
```

In [30]:
```
our_dataset['predict'] = prd
```

In [34]:
```
our_dataset.to_csv('logistic_regression_trained_on_outside.csv', index=False)
```

In [33]:
```
temp = pd.read_csv('baseline_clean_df.csv')
temp
```

Out[33]:

	Unnamed: 0	scene_name	scene_avg_p	scene_avg_a	scene_avg_d	scene_avg_blur	scene_avg
0	0	big_hero_60	64.405538	72.981351	82.268261	204.102944	
1	1	big_hero_61	76.658546	45.019868	93.268124	2058.214860	
2	2	big_hero_610	93.844222	48.897762	113.486587	319.563208	
3	3	big_hero_611	104.689522	68.900088	128.197153	344.318350	
4	4	big_hero_612	52.825514	48.692155	66.235224	302.668920	
5	5	big_hero_613	54.635902	56.085796	69.141541	373.148184	
6	6	big_hero_614	60.207244	39.892563	73.756332	187.768550	
7	7	big_hero_615	51.286053	48.994812	64.496347	81.434549	
8	8	big_hero_616	45.333909	38.411284	56.466676	110.447071	
9	9	big_hero_617	66.545265	37.953094	80.838276	609.955143	
10	10	big_hero_618	62.003990	58.110423	77.850133	297.200018	
11	11	big_hero_619	51.165063	48.256860	64.275012	475.489556	
12	12	big_hero_62	86.242043	28.622364	102.479761	595.473182	
13	13	big_hero_620	65.270362	51.280798	80.851837	300.750671	
14	14	big_hero_621	70.021676	55.197043	86.757769	88.788430	
15	15	big_hero_622	43.435414	49.524243	55.516054	294.189997	

	Unnamed: 0	scene_name	scene_avg_p	scene_avg_a	scene_avg_d	scene_avg_blur	scene_avg
16	16	big_hero_623	77.214954	2.256519	89.155313	955.737272	
17	17	big_hero_624	81.177807	16.607942	95.313364	284.872718	
18	18	big_hero_625	68.463573	8.830594	79.809820	665.750441	
19	19	big_hero_626	67.269475	11.599033	78.742681	217.413732	
20	20	big_hero_627	83.639257	16.274483	98.110386	538.016546	
21	21	big_hero_628	71.779220	11.691101	83.945388	919.960924	
22	22	big_hero_629	64.282734	12.975916	75.456829	567.952159	
23	23	big_hero_63	70.187601	46.987297	86.036241	179.866359	
24	24	big_hero_64	99.817869	21.813134	117.353930	1389.173305	
25	25	big_hero_65	45.452348	51.155879	58.019698	193.407722	
26	26	big_hero_66	63.912107	49.535743	79.093980	316.498497	
27	27	big_hero_67	63.395578	49.004431	78.440196	477.430012	
28	28	big_hero_68	99.640553	52.682391	120.581110	216.598992	
29	29	big_hero_69	72.928324	50.305583	89.560732	260.376936	
...	...	...	...	...	...	...	
150	150	wall_e_0	109.964774	5.142499	127.183908	210.344503	
151	151	wall_e_1	134.725784	12.771496	156.541492	1063.675243	

	Unnamed: 0	scene_name	scene_avg_p	scene_avg_a	scene_avg_d	scene_avg_blur	scene_avg
152	152	wall_e_10	83.493319	43.639831	100.984209	273.996571	
153	153	wall_e_11	101.908041	2.083827	117.567477	311.330457	
154	154	wall_e_12	131.064092	11.625181	152.198037	93.243649	
155	155	wall_e_13	112.984234	47.287686	135.345223	294.037656	
156	156	wall_e_14	61.531400	83.089274	80.082577	80.645540	
157	157	wall_e_15	38.170475	84.614697	53.354615	15.859611	
158	158	wall_e_16	44.637933	46.802282	56.598056	334.076365	
159	159	wall_e_17	48.400827	50.942584	61.390837	126.981297	
160	160	wall_e_18	50.750553	84.304011	67.804649	144.422954	
161	161	wall_e_19	71.001583	47.836532	87.067850	123.795181	
162	162	wall_e_2	128.117325	28.841856	150.718919	117.431503	
163	163	wall_e_20	43.134190	67.908255	57.212743	8.326760	
164	164	wall_e_21	49.672483	80.516533	66.142365	54.562706	
165	165	wall_e_22	59.196900	68.921784	75.819839	63.352845	
166	166	wall_e_23	24.881874	40.567980	33.158151	63.020314	
167	167	wall_e_24	24.359481	40.373340	32.535037	104.740910	
168	168	wall_e_25	30.307500	52.689583	40.752563	21.010125	

	Unnamed: 0	scene_name	scene_avg_p	scene_avg_a	scene_avg_d	scene_avg_blur	scene_avg
169	169	wall_e_26	154.245262	-4.729488	177.070653	1099.736528	
170	170	wall_e_27	115.650440	11.310007	134.415886	1437.663826	
171	171	wall_e_28	151.523748	-7.375126	173.643049	1265.153561	
172	172	wall_e_29	147.637915	-5.319448	169.397445	482.955341	
173	173	wall_e_3	123.088365	43.757472	146.586604	1225.754507	
174	174	wall_e_4	103.614057	65.383745	126.568007	1193.302027	
175	175	wall_e_5	126.631087	40.035603	150.251945	336.179835	
176	176	wall_e_6	112.835680	33.719842	133.666016	406.848319	
177	177	wall_e_7	91.598655	68.489974	113.078880	272.351328	
178	178	wall_e_8	84.844116	78.086689	106.368519	537.730281	
179	179	wall_e_9	109.652572	41.289332	130.842422	230.100643	

180 rows × 9 columns

In [ ]: