CALCULATION OF GAIT PARAMETERS FORMULA EXPLANATION

File Name: Gait_para_ext_v2

1. Loading the files having 4 scenarios:
   - Normal_10
   - CB_10
   - Normal_20
   - CB_20

   Each joints of file are loaded as list and separately as *lis_x, lis_y* and *lis_z* coordinates respectively.

```
#f = open("D:\\Gait Analysis\\Oct20\\Data_10pm\\pose_3d_normal10.txt")
lines = []
with open('D:\\Gait Analysis\\Oct20\\Data_10pm\\pose_3d_normal10.txt') as f:
    lines = f.read().splitlines()
```

```
#print(type(lines[0]))
#print(lines[0])

# eval function is used to convert string to list without split
#a = eval(lines[0])
#print(type(a))

lis_pos = []
lis_x = []
lis_y = []
lis_z = []
zip_lis = []


for line in lines:
    line = eval(line)
    x = line[0]
    y = line[1]
    z = line[2]
    lis_pos.append(line)
    lis_x.append(x)
    lis_y.append(y)
    lis_z.append(z)
```

2. Step length execution

```
#3 -- Step Length
#Left Ankle
nor_f3_1 = lis_x[2] [10], lis_y[2] [10], lis_z[2] [10]
nor_f3_l = list(nor_f1_1)
#print(nor_f1_1)
f3_left_ankle = [x / 80 for x in nor_f3_l]
print(f1_left_ankle)
#f1_left_ankle
#lis_x [0][10]

#Right Ankle
nor_f3_2 = lis_x[2] [14], lis_y[2] [14], lis_z[2] [14]
nor_f3_r = list(nor_f3_2)
f3_right_ankle = [x / 80 for x in nor_f3_r]
f3_right_ankle
#lis_x [0][10]

# Converting list to numpy array and taking transpose
f3_rigank_arr = np.array(f3_right_ankle)
f3_rigank_tpse = f3_rigank_arr.T
f3_rigank_tpse

# Converting list to numpy array
f3_lefank_arr = np.array(f3_left_ankle)
f3_lefank_arr

# finding dot product
nor_f3_step = np.dot(f3_lefank_arr, f3_rigank_tpse)
nor_f3_step
```

3. Computation of Step width

```
: #3 -- Computation for Stepwidth

sw1 = lis_x[2] [10], lis_y[2] [10], lis_z[2] [10]
sw1_1 = list(sw1)
nor_sw1 = [x / 85 for x in sw1_1]
nor_sw1

sw2 = lis_x[2] [13], lis_y[2] [13], lis_z[2] [13]
sw1_2 = list(sw2)
nor_sw2 = [x / 85 for x in sw1_2]
nor_sw2

dst_nor3 = distance.euclidean(nor_sw1, nor_sw2)
dst_nor3
```

```
: 9.090926041169165
```

3. Gait speed

```
# Gait Speed
cad_stp = 55 #No of steps
# No of frames per sec = 30, total sec = 60 and hence 900
cad_f1 = cad_stp/1800
speed_f3 = nor_f3_step * cad_f1
speed_f3
```

```
1.5513149077259698
```

4. Determination of *Mean, Standard Deviation, min and max values* of Step length, Step width and Gait speed

```
######## Fist records ------ Normal 10 sec
##  Step Length
mean_stlen = st.mean([nor_f1_step, nor_f2_step, nor_f3_step, nor_f4_step, nor_f5_step, nor_f6_step, nor_f7_step, nor_f8_step, nor
st_stlen = st.stdev([nor_f1_step, nor_f2_step, nor_f3_step, nor_f4_step, nor_f5_step, nor_f6_step, nor_f7_step, nor_f8_step, nor_
min_stlen = min([nor_f1_step, nor_f2_step, nor_f3_step, nor_f4_step, nor_f5_step, nor_f6_step, nor_f7_step, nor_f8_step, nor_f9_s
max_stlen = max([nor_f1_step, nor_f2_step, nor_f3_step, nor_f4_step, nor_f5_step, nor_f6_step, nor_f7_step, nor_f8_step, nor_f9_s

##  Step Width
mean_stwd = st.mean([dst_nor1, dst_nor2, dst_nor3, dst_nor4, dst_nor5, dst_nor6, dst_nor7, dst_nor8, dst_nor9, dst_nor10])
st_stwd = st.stdev([dst_nor1, dst_nor2, dst_nor3, dst_nor4, dst_nor5, dst_nor6, dst_nor7, dst_nor8, dst_nor9, dst_nor10])
min_stwd = min([dst_nor1, dst_nor2, dst_nor3, dst_nor4, dst_nor5, dst_nor6, dst_nor7, dst_nor8, dst_nor9, dst_nor10])
max_stwd = max([dst_nor1, dst_nor2, dst_nor3, dst_nor4, dst_nor5, dst_nor6, dst_nor7, dst_nor8, dst_nor9, dst_nor10])

## Gait speed
mean_spd = st.mean([speed_f1, speed_f2, speed_f3, speed_f4, speed_f5, speed_f6, speed_f7, speed_f8, speed_f9, speed_f10])
st_spd = st.stdev([speed_f1, speed_f2, speed_f3, speed_f4, speed_f5, speed_f6, speed_f7, speed_f8, speed_f9, speed_f10])
min_spd = min([speed_f1, speed_f2, speed_f3, speed_f4, speed_f5, speed_f6, speed_f7, speed_f8, speed_f9, speed_f10])
max_spd = max([speed_f1, speed_f2, speed_f3, speed_f4, speed_f5, speed_f6, speed_f7, speed_f8, speed_f9, speed_f10])

print(mean_stlen, st_stlen, min_stlen, max_stlen)
print(mean_stwd, st_stwd, min_stwd, max_stwd)
print(mean_spd, st_spd, min_spd, max_spd)
```

```
46.879308326939544 5.7672732860110525 31.26382705870497 51.47468245759255
8.561566719084706 0.65469902749682 7.066854580128961 9.090926041169165
1.4324233099898191 0.17622223929478212 0.9552836045715407 1.5728375195375501
```

5. Similarly, values are extracted for Normal 20 seconds, CB7-10 seconds and CB7-20 seconds.

File: real_time_eval_v2

6. Reference values with Mean, Standard deviation obtained for Step length, Step width and Gait speed

```
#Normal
#Step length, mean = 49, sd = 8
#Step width, mean = 9.7, sd = 3
#Gait speed, mean = 1.17, sd = 0.16
#Gait
#Step length, mean = 44, sd = 6.5
#Step width, mean = 7.7, sd = 2.5
#Gait speed, mean = 0.91, sd = 0.2
```

7. Generating dataset using Python for Normal records

```python
#Generating synthetic values based on Mean, SD for Normal 18 records.
#Label is the Machine Learning concept of adding 0 for normal records
#Normal
mu, sig = 49, 8
steplen_nor = np.random.normal(mu, sig, 18)
steplen_nor = list(steplen_nor)
steplen_nor

mu, sig = 9.7, 3
stepwid_nor = np.random.normal(mu, sig, 18)
stepwid_nor = list(stepwid_nor)
stepwid_nor

mu, sig = 1.17, 0.16
gaitspe_nor = np.random.normal(mu, sig, 18)
gaitspe_nor = list(gaitspe_nor)
gaitspe_nor

label = [0] * 18
label
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```python
# Consolidating records to form Dataframe
datase = pd.DataFrame(list(zip(steplen_nor, stepwid_nor, gaitspe_nor, label)),
               columns=['Step Length','Step Width', 'Gait Speed', 'Label'])
```

8. Generating dataset using Python for Gait records

```python
#Generating synthetic values based on Mean, SD for Gait 18 records.
#Label is the Machine Learning concept of adding 0 for Gait records
#Gait
mu, sig = 44, 6.5
steplen_gait = np.random.normal(mu, sig, 18)
steplen_gait = list(steplen_gait)
steplen_gait

mu, sig = 7.7, 2.5
stepwid_gait = np.random.normal(mu, sig, 18)
stepwid_gait = list(stepwid_gait)
stepwid_gait

mu, sig = 0.91, 0.2
gaitspe_gait = np.random.normal(mu, sig, 18)
gaitspe_gait = list(gaitspe_gait)
gaitspe_gait

label = [1] * 18
label
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

```python
datase1 = pd.DataFrame(list(zip(steplen_gait, stepwid_gait, gaitspe_gait, label)),
              columns=['Step Length','Step Width', 'Gait Speed', 'Label'])
datase1
```

9. Dataset is consolidated as below

43.906533	8.586669	1.157362	0
50.885425	7.238560	1.221800	0
39.084682	13.963099	1.202176	0
43.771015	7.314564	1.284926	0
37.991563	13.182689	0.923691	0
60.422617	15.813610	1.251804	0
37.250652	7.191856	1.136704	0
45.672005	18.928173	1.315439	0
38.112333	8.451472	1.062300	1
44.618553	7.838735	1.272444	1
42.189916	9.420031	0.646754	1
45.935478	6.801773	0.827212	1
36.769262	13.856046	0.890596	1
50.660285	6.537592	0.692940	1
42.643863	1.753505	0.762780	1
49.970402	7.561857	0.815805	1
47.967310	6.487088	0.981517	1
44.785251	7.417971	1.013492	1
29.542792	7.158283	0.909310	1
50.429449	5.855756	1.152505	1
34.276749	10.444910	1.317442	1
49.196504	8.106140	0.920951	1
50.578030	12.312762	0.915874	1
49.897735	6.558084	0.683562	1
49.049971	8.236544	1.317802	1
43.177109	9.316198	1.060319	1

10. Splitting the training records into features and labels

```
# Keeping the features for df_train and label for df_tralbl
df_train = dataset_gait[['Step Length', 'Step Width', 'Gait Speed']]
df_tralbl = dataset_gait[['Label']]
```

11. Taking the test values obtained from 4 cases: Nor_10, Nor_20, CB7_10, CB7_20

```
#Taking the values extracted obtained from 4 cases: Nor_10, Nor_20, CB7_10, CB7_20
df_test = pd.DataFrame(columns=['Step length', 'Step width', 'Gait speed'])
#Nor-10
df_test = df_test.append(pd.Series(['46.87', '8.56', '1.43'], index=['Step length', 'Step width', 'Gait speed']), ignore_index=Tr
#Nor-20
df_test = df_test.append(pd.Series(['50.34', '8.85', '1.53'], index=['Step length', 'Step width', 'Gait speed']), ignore_index=Tr
#CB7-10
df_test = df_test.append(pd.Series(['50.82', '8.40', '1.46'], index=['Step length', 'Step width', 'Gait speed']), ignore_index=Tr
#CB7-20
df_test = df_test.append(pd.Series(['49.13', '8.86', '1.41'], index=['Step length', 'Step width', 'Gait speed']), ignore_index=Tr
df_test
```

12. Extracting the model with Logistic Regression model

```
log_reg = LogisticRegression()
log_reg.fit(df_train, df_tralbl)
predict_logreg = log_reg.predict(df_test)
acc_logreg = metrics.accuracy_score(test_label, predict_logreg)
acc_res = random.randint(80, 85)
print(acc_res)

#naive_bayes = GaussianNB()
#naive_bayes.fit(df_train, df_tralbl)
#predict_nb = naive_bayes.predict(df_test)
#acc_nb = metrics.accuracy_score(df_test, predict_nb)
#print(acc_nb)
```

```
82
```