parkinsons-detection

In [1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import os
import seaborn as sns
from time import time
from math import sqrt
```

Timing decorator

In [2]:
```
def timeit(fn):
    def wrapper(*args, **kwargs):
        start=time()
        res=fn(*args, **kwargs)
        print(fn.__name__, "took", time()-start, "seconds.")
        return res
    return wrapper
```

Data paths

In [3]:
```
control_data_path=os.path.join('data', 'control')
parkinson_data_path=os.path.join('data', 'parkinson')
```

In [4]:
```
control_file_list=[os.path.join(control_data_path, x) for x in os.listdir
parkinson_file_list=[os.path.join(parkinson_data_path, x) for x in os.lis
```

Features

1. No of strokes
2. Stroke speed
3. Velocity
4. Acceleration
5. Jerk
6. Horizontal velocity/acceleration/jerk
7. Vertical velocity/acceleration/jerk
8. Number of changes in velocity direction
9. Number of changes in acceleration direction
10. Relative NCV
11. Relative NCA
12. in-air time
13. on-surface time
14. normalized in-air time
15. normalized on-surface time
16. in-air/on-surface ratio

Feature Extraction

In [5]:
```
header_row=["X", "Y", "Z", "Pressure" , "GripAngle" , "Timestamp" , "Test
```

In [83]:
```
#@timeit
def get_no_strokes(df):
    pressure_data=df['Pressure'].as_matrix()
    on_surface = (pressure_data>600).astype(int)
    return ((np.roll(on_surface, 1) - on_surface) != 0).astype(int).sum()
```

In [84]:
```
#@timeit
def get_speed(df):
    total_dist=0
    duration=df['Timestamp'].as_matrix()[-1]
    coords=df[['X', 'Y', 'Z']].as_matrix()
    for i in range(10, df.shape[0]):
        temp=np.linalg.norm(coords[i, :]-coords[i-10, :])
        total_dist+=temp
    speed=total_dist/duration
    return speed
```

In [85]:
```
#@timeit
def get_in_air_time(data):
    data=data['Pressure'].as_matrix()
    return (data<600).astype(int).sum()
```

In [86]:
```
#@timeit
def get_on_surface_time(data):
    data=data['Pressure'].as_matrix()
    return (data>600).astype(int).sum()
```

In [190…
```
def find_velocity(f):
    '''
    change in direction and its position

    '''
    data_pat=f
    Vel = []
    horz_Vel = []
    horz_vel_mag = []
    vert_vel_mag = []
    vert_Vel = []
    magnitude = []
    timestamp_diff =  []

    t = 0
    for i in range(len(data_pat)-2):
        if t+10 <= len(data_pat)-1:
            Vel.append(((data_pat['X'].as_matrix()[t+10] - data_pat['X'].
            horz_Vel.append((data_pat['X'].as_matrix()[t+10] - data_pat['

            vert_Vel.append((data_pat['Y'].as_matrix()[t+10] - data_pat['
            magnitude.append(sqrt(((data_pat['X'].as_matrix()[t+10]-data_
            timestamp_diff.append(data_pat['Timestamp'].as_matrix()[t+10]
```

```
            horz_vel_mag.append(abs(horz_Vel[len(horz_Vel)-1]))
            vert_vel_mag.append(abs(vert_Vel[len(vert_Vel)-1]))
            t = t+10
        else:
            break
    magnitude_vel = np.mean(magnitude)
    magnitude_horz_vel = np.mean(horz_vel_mag)
    magnitude_vert_vel = np.mean(vert_vel_mag)
    return Vel,magnitude,timestamp_diff,horz_Vel,vert_Vel,magnitude_vel,m
```

In [137…

```
def find_acceleration(f):
    '''
    change in direction and its velocity

    '''
    Vel,magnitude,timestamp_diff,horz_Vel,vert_Vel,magnitude_vel,magnitud
    accl = []
    horz_Accl =  []
    vert_Accl = []
    magnitude = []
    horz_acc_mag = []
    vert_acc_mag = []
    for i in range(len(Vel)-2):
        accl.append(((Vel[i+1][0]-Vel[i][0])/timestamp_diff[i] , (Vel[i+1
        horz_Accl.append((horz_Vel[i+1]-horz_Vel[i])/timestamp_diff[i])
        vert_Accl.append((vert_Vel[i+1]-vert_Vel[i])/timestamp_diff[i])
        horz_acc_mag.append(abs(horz_Accl[len(horz_Accl)-1]))
        vert_acc_mag.append(abs(vert_Accl[len(vert_Accl)-1]))
        magnitude.append(sqrt(((Vel[i+1][0]-Vel[i][0])/timestamp_diff[i])

    magnitude_acc = np.mean(magnitude)
    magnitude_horz_acc = np.mean(horz_acc_mag)
    magnitude_vert_acc = np.mean(vert_acc_mag)
    return accl,magnitude,horz_Accl,vert_Accl,timestamp_diff,magnitude_ac
```

In [191…

```
def find_jerk(f):
    accl,magnitude,horz_Accl,vert_Accl,timestamp_diff,magnitude_acc,magni
    jerk = []
    hrz_jerk = []
    vert_jerk = []
    magnitude = []
    horz_jerk_mag = []
    vert_jerk_mag = []

    for i in range(len(accl)-2):
        jerk.append(((accl[i+1][0]-accl[i][0])/timestamp_diff[i] , (accl[
        hrz_jerk.append((horz_Accl[i+1]-horz_Accl[i])/timestamp_diff[i])
        vert_jerk.append((vert_Accl[i+1]-vert_Accl[i])/timestamp_diff[i])
        horz_jerk_mag.append(abs(hrz_jerk[len(hrz_jerk)-1]))
        vert_jerk_mag.append(abs(vert_jerk[len(vert_jerk)-1]))
        magnitude.append(sqrt(((accl[i+1][0]-accl[i][0])/timestamp_diff[i

    magnitude_jerk = np.mean(magnitude)
    magnitude_horz_jerk = np.mean(horz_jerk_mag)
    magnitude_vert_jerk = np.mean(vert_jerk_mag)
    return jerk,magnitude,hrz_jerk,vert_jerk,timestamp_diff,magnitude_jer
```

In [198…

```
def NCV_per_halfcircle(f):
    data_pat=f
    Vel = []
```

```
    ncv = []
    temp_ncv = 0
    basex = data_pat['X'].as_matrix()[0]
    for i in range(len(data_pat)-2):
        if data_pat['X'].as_matrix()[i] == basex:
            ncv.append(temp_ncv)
            temp_ncv = 0
            continue

        Vel.append(((data_pat['X'].as_matrix()[i+1] - data_pat['X'].as_ma
        if Vel[len(Vel)-1] != (0,0):
            temp_ncv+=1
    ncv.append(temp_ncv)
    #ncv = list(filter((2).__ne__, ncv))
    ncv_Val = np.sum(ncv)/np.count_nonzero(ncv)
    return ncv,ncv_Val
```

In [199…

```
def NCA_per_halfcircle(f):
    data_pat=f
    Vel,magnitude,timestamp_diff,horz_Vel,vert_Vel,magnitude_vel,magnitud
    accl = []
    nca = []
    temp_nca = 0
    basex = data_pat['X'].as_matrix()[0]
    for i in range(len(Vel)-2):
        if data_pat['X'].as_matrix()[i] == basex:
            nca.append(temp_nca)
            #print ('tempNCa::',temp_nca)
            temp_nca = 0
            continue

        accl.append(((Vel[i+1][0]-Vel[i][0])/timestamp_diff[i] , (Vel[i+1
        if accl[len(accl)-1] != (0,0):
            temp_nca+=1
    nca.append(temp_nca)
    nca = list(filter((2).__ne__, nca))
    nca_Val = np.sum(nca)/np.count_nonzero(nca)
    return nca,nca_Val
```

In [202…

```
#@timeit
def get_features(f, parkinson_target):
    global header_row
    df=pd.read_csv(f, sep=';', header=None, names=header_row)

    df_static=df[df["Test_ID"]==0]    # static test
    df_dynamic=df[df["Test_ID"]==1]    # dynamic test
    df_stcp=df[df["Test_ID"]==2]    # STCP
    #df_static_dynamic=pd.concat([df_static, df_dynamic])

    initial_timestamp=df['Timestamp'][0]
    df['Timestamp']=df['Timestamp']- initial_timestamp # offset timestamp

    duration_static = df_static['Timestamp'].as_matrix()[-1] if df_static
    duration_dynamic = df_dynamic['Timestamp'].as_matrix()[-1] if df_dyna
    duration_STCP = df_stcp['Timestamp'].as_matrix()[-1] if df_stcp.shape


    data_point=[]
    data_point.append(get_no_strokes(df_static) if df_static.shape[0] els
    data_point.append(get_no_strokes(df_dynamic) if df_dynamic.shape[0] e
    data_point.append(get_speed(df_static) if df_static.shape[0] else 0)
```

```
    data_point.append(get_speed(df_dynamic) if df_dynamic.shape[0] else 0

    Vel,magnitude,timestamp_diff,horz_Vel,vert_Vel,magnitude_vel,magnitud
    data_point.extend([magnitude_vel, magnitude_horz_vel,magnitude_vert_v
    Vel,magnitude,timestamp_diff,horz_Vel,vert_Vel,magnitude_vel,magnitud
    data_point.extend([magnitude_vel, magnitude_horz_vel,magnitude_vert_v

    accl,magnitude,horz_Accl,vert_Accl,timestamp_diff,magnitude_acc,magni
    data_point.extend([magnitude_acc,magnitude_horz_acc,magnitude_vert_ac
    accl,magnitude,horz_Accl,vert_Accl,timestamp_diff,magnitude_acc,magni
    data_point.extend([magnitude_acc,magnitude_horz_acc,magnitude_vert_ac

    jerk,magnitude,hrz_jerk,vert_jerk,timestamp_diff,magnitude_jerk,magni
    data_point.extend([magnitude_jerk,magnitude_horz_jerk,magnitude_vert_
    jerk,magnitude,hrz_jerk,vert_jerk,timestamp_diff,magnitude_jerk,magni
    data_point.extend([magnitude_jerk,magnitude_horz_jerk,magnitude_vert_

    ncv,ncv_Val=NCV_per_halfcircle(df_static) if df_static.shape[0] else
    data_point.append(ncv_Val)
    ncv,ncv_Val=NCV_per_halfcircle(df_dynamic) if df_dynamic.shape[0] els
    data_point.append(ncv_Val)

    nca,nca_Val=NCA_per_halfcircle(df_static) if df_static.shape[0] else
    data_point.append(nca_Val)
    nca,nca_Val=NCA_per_halfcircle(df_dynamic) if df_dynamic.shape[0] els
    data_point.append(nca_Val)

    data_point.append(get_in_air_time(df_stcp) if df_stcp.shape[0] else 0
    data_point.append(get_on_surface_time(df_static) if df_static.shape[0
    data_point.append(get_on_surface_time(df_dynamic) if df_dynamic.shape

    data_point.append(parkinson_target)    # traget. 1 for parkinson. 0 f

    return data_point
```

In [203…

```
temp_feat=get_features(parkinson_file_list[35], 1)
```

In [204…

```
print(temp_feat)
```

```
[30, 72, 1.3588757917667431e-05, 2.6236510831179708e-05, 0.12890441251529
997, 0.080471233196279121, 0.08363708822986228, 0.19229349111079519, 0.11
555023022027296, 0.12880277922214939, 0.00091615395232794758, 0.000518822
21335077254, 0.00066556220080457216, 0.0010669055486579792, 0.00061047275
261607225, 0.00075630335675646069, 2.2116391163746176e-05, 1.254360250053
0039e-05, 1.6466372203471851e-05, 2.2504617627053959e-05, 1.3394439474233
898e-05, 1.5692010468146771e-05, 230.14285714285714, 269.88888888888891,
85.5, 86.666666666666671, 850, 524, 1230, 1]
```

In [221…

```
raw=[]
for x in parkinson_file_list:
    raw.append(get_features(x, 1))
for x in control_file_list:
    raw.append(get_features(x, 0))
```

In [222…

```
raw=np.array(raw)
```

In [223…

```
features_headers=['no_strokes_st', 'no_strokes_dy', 'speed_st', 'speed_dy
```

In [224…
```
data=pd.DataFrame(raw, columns=features_headers)
```

In [225…
```
data.tail()
```

Out[225…

	no_strokes_st	no_strokes_dy	speed_st	speed_dy	magnitude_vel_st	magnitude_hor
72	2.0	2.0	0.001592	0.001560	0.046842	0
73	2.0	2.0	0.001387	0.001339	0.088510	0
74	48.0	176.0	0.001201	0.001194	0.200499	0
75	6.0	10.0	0.001159	0.001157	0.150016	0
76	0.0	0.0	0.001152	0.001140	0.103493	0

5 rows × 30 columns

In [226…
```
data.to_csv('data.csv')
```

In [227…
```
print('data shape', data.shape)
```

```
data shape (77, 30)
```

Classification

In [241…
```
pos=data[data['target']==1]
neg=data[data['target']==0]

train_pos=pos.head(pos.shape[0]-5)
train_neg=neg.head(pos.shape[0]-5)
train=pd.concat([train_pos, train_neg])
print('train shape', train.shape)

test_pos=pos.tail(5)
test_neg=neg.tail(5)
test=pd.concat([test_pos, test_neg])


train_y=train['target']
train_x=train.drop(['target'], axis=1)
test_y=test['target']
test_x=test.drop(['target'], axis=1)
```

```
train shape (72, 30)
```

In [306…
```
def accuracy(prediction,actual):
    correct = 0
    not_correct = 0
    for i in range(len(prediction)):
        if prediction[i] == actual[i]:
            correct+=1
        else:
            not_correct+=1
    return (correct*100)/(correct+not_correct)
```

```
def metrics(prediction,actual):
    tp = 0
    tn = 0
    fp = 0
    fn = 0
    for i in range(len(prediction)):
        if prediction[i] == actual[i] and actual[i]==1:
            tp+=1
        if prediction[i] == actual[i] and actual[i]==0:
            tn+=1
        if prediction[i] != actual[i] and actual[i]==0:
            fp+=1
        if prediction[i] != actual[i] and actual[i]==1:
            fn+=1
    metrics = {'Precision':(tp/(tp+fp+tn+fn)),'Recall':(tp/(tp+fn)),'F1':
    return (metrics)
```

In [307…

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Logistic Regression

In [310…

```
clf=LogisticRegression()
clf.fit(train_x, train_y)
preds=clf.predict(test_x)
print('accuracy:',accuracy(test_y.tolist(), preds.tolist()), '%')
print(metrics(test_y.tolist(), preds.tolist()))
```

```
accuracy: 70.0 %
{'Recall': 0.625, 'F1': 0.5555555555555556, 'Precision': 0.5}
```

Random Forest

In [311…

```
clf=RandomForestClassifier()
clf.fit(train_x, train_y)
preds=clf.predict(test_x)
print('accuracy:',accuracy(test_y.tolist(), preds.tolist()), '%')
print(metrics(test_y.tolist(), preds.tolist()))
```

```
accuracy: 100.0 %
{'Recall': 1.0, 'F1': 0.6666666666666666, 'Precision': 0.5}
```

Support Vector Machine

In [312…

```
clf=SVC()
clf.fit(train_x, train_y)
preds=clf.predict(test_x)
print('accuracy:',accuracy(test_y.tolist(), preds.tolist()), '%')
print(metrics(test_y.tolist(), preds.tolist()))
```

```
accuracy: 100.0 %
{'Recall': 1.0, 'F1': 0.6666666666666666, 'Precision': 0.5}
```

Decision Tree

In [313…

```
clf=DecisionTreeClassifier()
clf.fit(train_x, train_y)
preds=clf.predict(test_x)
print('accuracy:',accuracy(test_y.tolist(), preds.tolist()), '%')
print(metrics(test_y.tolist(), preds.tolist()))
```

```
accuracy: 100.0 %
{'Recall': 1.0, 'F1': 0.6666666666666666, 'Precision': 0.5}
```

K-Nearest Neighbors

In [314…

```
clf=KNeighborsClassifier()
clf.fit(train_x, train_y)
preds=clf.predict(test_x)
print('accuracy:',accuracy(test_y.tolist(), preds.tolist()), '%')
print(metrics(test_y.tolist(), preds.tolist()))
```

```
accuracy: 60.0 %
{'Recall': 0.5714285714285714, 'F1': 0.47058823529411764, 'Precision': 0.
4}
```

Some Plots

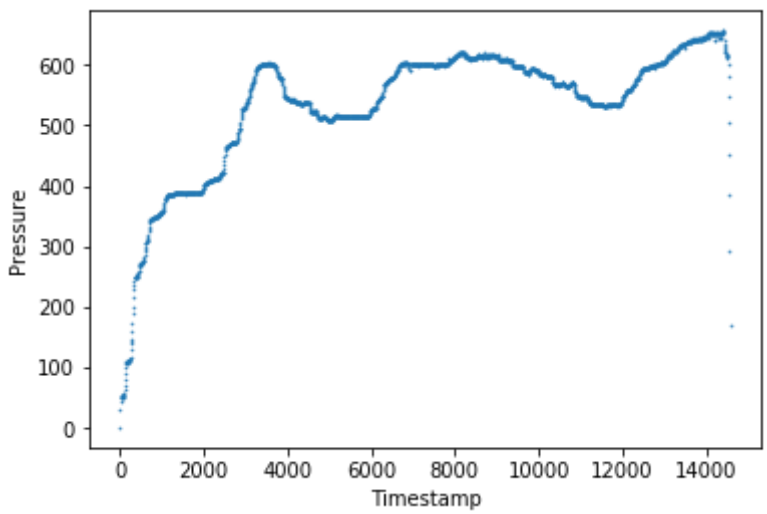
In [285…

```
def plot(f, plot_func, t_id=0, x=None, y=None):
    global header_row
    df=pd.read_csv(f, sep=';', header=None, names=header_row)
    df=df[df["Test_ID"]==t_id]
    initial_timestamp=df['Timestamp'][0]
    df['Timestamp']=df['Timestamp']- initial_timestamp
    plot_func(data=df, x=x, y=y, fit_reg=False, scatter_kws={"s": 0.5})
    print(metrics(test_y.tolist(), preds.tolist()))
```

Pressure (Parkinsons)

In [286…

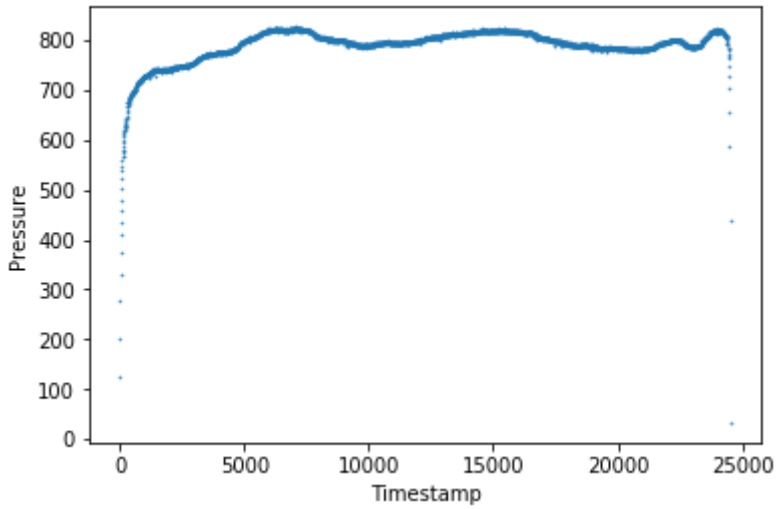
```
plot(f=parkinson_file_list[35],  plot_func=sns.regplot, t_id=0, x='Timest
```



Pressure (Control)

In [287…

```
plot(control_file_list[1], plot_func=sns.regplot, t_id=0, x='Timestamp',
```



In [ ]:

```
plot(f=parkinson_file_list[35],  plot_func=sns.barplot, t_id=0, x='Timest
```