linear_regression_XYZ_MET-walking_old

January 12, 2021

[1]:
```
from helpers import pandas_helper as pdh
from helpers import math_helper as mth
from sensors.activpal import *
from utils import read_functions
from scipy.stats import linregress
import math
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime

activpal = Activpal()
activity_focus = 'lopen'
```

0.0.1 Method to retrieve DataFrame containing all information for regression

[2]:
```
def get_regression_df(respondent, activity):
    start, stop = get_timestamps(respondent, activity)

    # read in all dataframes necessary
    respondents_df = pdh.read_csv_respondents()
    res_number = get_respondent_number(respondent)
    vyntus_df, min_index, max_index = get_vyntus_df(respondent,␣
 ↪respondents_df['gewicht'][res_number], start, stop)
    raw_df = get_raw_df(respondent, min_index, max_index)

    # add met and mag acc to new dataframe
    new_df = pd.DataFrame(index=raw_df.index)
    new_df['mean_met'] = vyntus_df['met']
    new_df['sum_mag_acc'] = raw_df['mag_acc']

    # add features to new dataframe
    new_df['length'] = respondents_df['lengte'][res_number] / 100
    new_df['weight'] = respondents_df['gewicht'][res_number]
    new_df['speed'] = raw_df['speed']
    new_df['bmi'] = mth.calculate_bmi(new_df['weight'], new_df['length'])
```

```
    new_df['gender'] = int(respondents_df['geslacht'][res_number].
 ↪replace('vrouw',str(0)).replace('man', str(1)))
    new_df['age_cat'] = respondents_df['leeftijdscategorie'][res_number]
    convert_age_to_number(new_df, "age_cat")
    new_df['sporter'] = respondents_df['sporter'][res_number]
    new_df['exercise_guideline_compliance'] = int(respondents_df['voldoet aan␣
 ↪beweegrichtlijn 2nee17'][res_number].replace('ja', '1').replace('nee', '0'))
    new_df['does_muscle_bone_exercises'] = int(respondents_df['voldoet aan␣
 ↪richtlijn bot en spierversterkende activiteiten'][res_number].replace('ja',␣
 ↪'1').replace('nee', '0'))
    new_df['balance_guideline_compliance'] = int(respondents_df['voldoet aan␣
 ↪richtlijn balansoefeningen'][res_number].replace('ja', '1').replace('nee',␣
 ↪'0'))

    return new_df


def get_regression_dfs(respondents, activity):
    all_df = pd.DataFrame(index=pd.to_datetime([]))

    for cor in respondents:
        df = get_regression_df(cor, activity)
        print(cor + ' - ' + str(df['sum_mag_acc'].corr(df['mean_met'])))
        all_df = pd.concat([all_df, df])


    all_df.sort_index(inplace=True)
    print(all_df['sum_mag_acc'].corr(all_df['mean_met']))
    return all_df
```

0.0.2 Helper method, returning start and stop timestamps for an activity

[3]:
```
def get_timestamps(respondent, activity):
    activities_df = read_functions.read_activities(respondent)
    start = activities_df.loc[activity].start
    stop = activities_df.loc[activity].stop

    return (start, stop)
```

[4]:
```
def get_respondent_number(respondent):
    if ('BMR0' in respondent):
        return int(respondent.replace('BMR0', ''))

    if ('BMR' in respondent):
        return int(respondent.replace('BMR', ''))
```

0.0.3 Helper method, returning Vyntus (lab data)

The DataFrame contains MET and other information necessary for regression

The DataFrame is resampled to minutes by mean

[5]:
```
def get_vyntus_df(respondent, weight, start, stop):
    vyntus_df = pdh.read_csv_vyntus(respondent)
    mask = (vyntus_df.index >= start) & (vyntus_df.index < stop)
    vyntus_df = vyntus_df.loc[mask]

    min_index = vyntus_df.index.min()
    max_index = vyntus_df.index.max()

    vyntus_df['vyn_VO2'] = [float(vo2.replace(',', '.')) if type(vo2) == str␣
↪else vo2 for vo2 in vyntus_df['vyn_VO2']]
    vyntus_df['met'] = mth.calculate_met(vyntus_df['vyn_VO2'], weight)

    vyntus_df = vyntus_df.resample('60s').mean()[:-1]

    return (vyntus_df, min_index, max_index)
```

0.0.4 Helper method, returning Activpal20

The DataFrame contains sumation of magnitude of acceleration

The DataFrame is resampled to minutes by sumation

[6]:
```
def get_raw_df(respondent, start, stop):
    df = activpal.read_data(respondent, start, stop)

    mask = (df.index >= start) & (df.index < stop)
    df = df.loc[mask]

    df = df[['pal_accX', 'pal_accY', 'pal_accZ']].apply(mth.convert_value_to_g)
    df['mag_acc'] = mth.to_mag_acceleration(df['pal_accX'], df['pal_accY'],␣
↪df['pal_accZ'])
    df['speed'] = get_speed(df['pal_accX'], df['pal_accY'], df['pal_accZ'])

    df = df.resample('60s').agg({'mag_acc':'sum', 'speed':'mean'})[:-1]

    return df
```

0.0.5 Helper method, from Colin

[7]:
```
def convert_age_to_number(dataframe, age_category):
    age_convertion = {
        age_category: {
            "15-19": 0, "20-24": 1, "25-29": 2, "30-34": 3,
            "35-39": 4,  "40-44": 5, "45-49": 6, "50-54": 7,
            "55-59": 8,  "60-64": 9, "65-69": 10,  "70-74": 11, "75-79": 12
        }
    }
    return dataframe.replace(age_convertion, inplace=True)
```

0.0.6 Method for speed, from Adnan

[8]:
```
import scipy.integrate as it

def get_speed(x_acc, y_acc, z_acc):

    activpal_time = 0.05

    #x_vel = np.array(it.cumtrapz(x_acc, dx=activpal_time, initial=0 ))
    #y_vel = np.array(it.cumtrapz(y_acc, dx=activpal_time, initial=0 ))
    #z_vel = np.array(it.cumtrapz(z_acc, dx=activpal_time, initial=0 ))

    x_vel = np.array([sum(x_acc[:i]) * activpal_time for i in␣
 ↪range(len(x_acc))])
    y_vel = np.array([sum(y_acc[:i]) * activpal_time for i in␣
 ↪range(len(y_acc))])
    z_vel = np.array([sum(z_acc[:i]) * activpal_time for i in␣
 ↪range(len(z_acc))])


    return np.sqrt(x_vel ** 2 + y_vel **2 + z_vel ** 2)
```

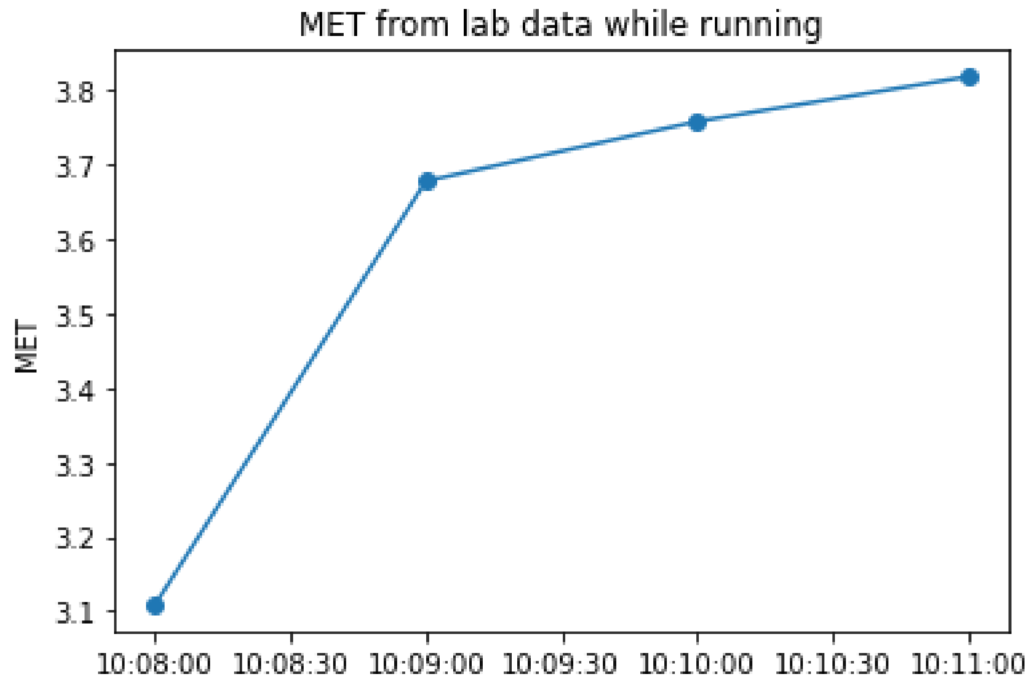
0.0.7 Helper methods to plot various graphs

[9]:
```
def plot_met(met, title = 'MET from lab data while running'):
    plt.ylabel('MET')
    plt.plot(met, marker = 'o')
    plt.title(title)

def plot_mag_acc(mag_acc, title = 'Accelerometer data while running'):
    plt.ylabel('Sum magnitude acceleration, g')
    plt.plot(mag_acc, marker = 'o')
```

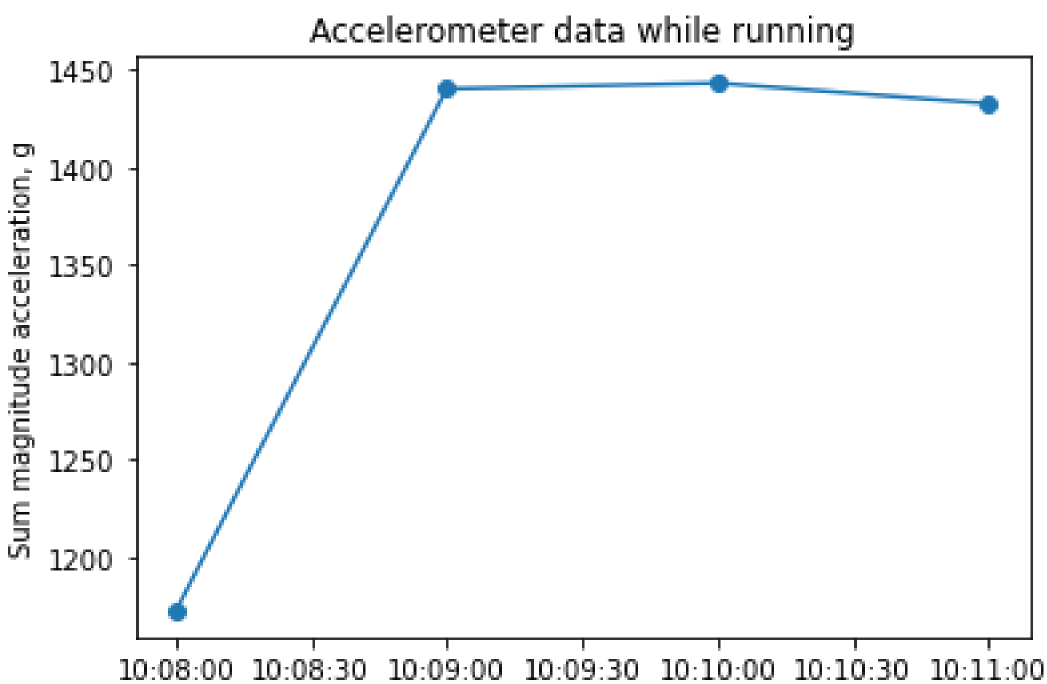
```
    plt.title(title)

def plot_lin_reg(x, y, xlabel = 'acceleration, g', ylabel = 'MET'):
    linreg = linregress(x, y)
    fx = np.array([x.min(), x.max()])
    fy = linreg.intercept + linreg.slope * fx

    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.plot(x, y, 'o')
    plt.plot(fx, fy, '-')

def plot_pred_truth(pred_y, valid_y, title):
    plt.plot(range(len(pred_y)), pred_y, label='prediction', marker='o')
    plt.plot(range(len(valid_y)),valid_y,label='ground_truth', marker='o')

    plt.title(title)
    plt.ylabel('met')
    plt.xlabel('Prediction Number')
    plt.legend()
```

0.0.8 Single linear regression running for one respondent

[10]:
```
res30 = 'BMR030'
df30 = get_regression_df(res30, activity_focus)
plot_met(df30['mean_met'])
```

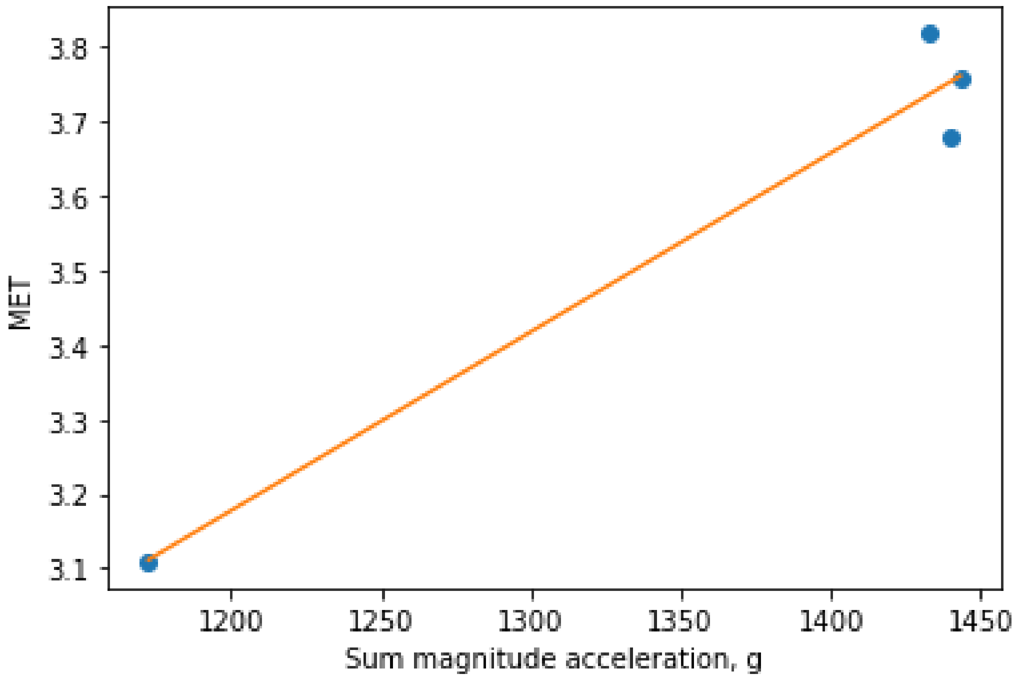


[11]:
```
plot_mag_acc(df30['sum_mag_acc'])
```



[12]:
```
print('r = ' + str(df30['sum_mag_acc'].corr(df30['mean_met'])))
plot_lin_reg(df30['sum_mag_acc'], df30['mean_met'], xlabel='Sum magnitude␣
↪acceleration, g')
```

```
r = 0.9800959338711214
```

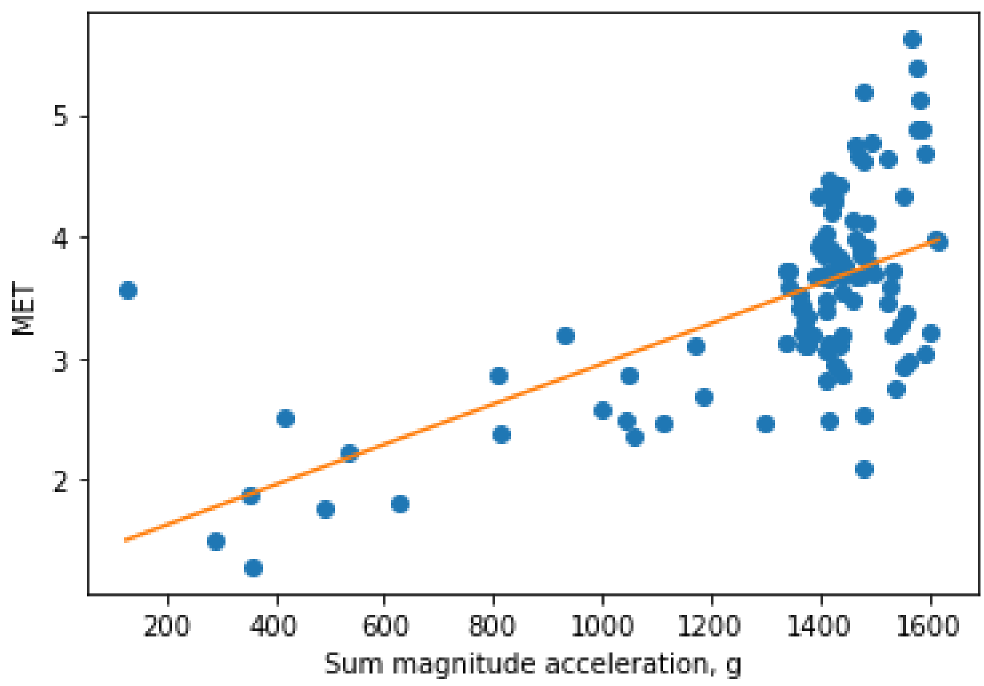


0.0.9 Single linear regression running for all respondents

[13]:
```
respondents = ['BMR002', 'BMR011', 'BMR012', 'BMR014', 'BMR018', 'BMR030',␣
↪'BMR031', 'BMR032', 'BMR033', 'BMR034','BMR036', 'BMR040', 'BMR041',␣
↪'BMR042', 'BMR043', 'BMR044', 'BMR052', 'BMR053', 'BMR055', 'BMR058',␣
↪'BMR064', 'BMR098']
test_respondents = ['BMR004', 'BMR008', 'BMR097']

all_df = get_regression_dfs(respondents, activity_focus)
test_df = get_regression_dfs(test_respondents, activity_focus)
plot_lin_reg(all_df['sum_mag_acc'], all_df['mean_met'], xlabel='Sum magnitude␣
↪acceleration, g')
```

```
BMR002 - 0.9197559831048818
BMR011 - 0.8859527619029444
BMR012 - 0.8304918100757134
BMR014 - 0.9474332896505901
BMR018 - 0.9285261149857478
```

```
BMR030 - 0.9800959338711214
BMR031 - 0.9946153473325094
BMR032 - 0.7489481734600001
BMR033 - 0.975990870349137
BMR034 - 0.8733365666574373
BMR036 - 0.864149472357056
BMR040 - 0.8655196433457287
BMR041 - 0.5723645775588121
BMR042 - 0.8324866410264637
BMR043 - 0.8518369189687028
BMR044 - 0.9562395634424609
BMR052 - 0.9883095668272954
BMR053 - 0.9296306478619453
BMR055 - 0.9053540009019323
BMR058 - 0.9336722955817809
BMR064 - 0.7746981146067212
BMR098 - 0.9404355640997635
0.6168424319085747
BMR004 - 0.9623798523019808
BMR008 - 0.9761865073645364
BMR097 - 0.8289528602677096
0.5512918834650794
```



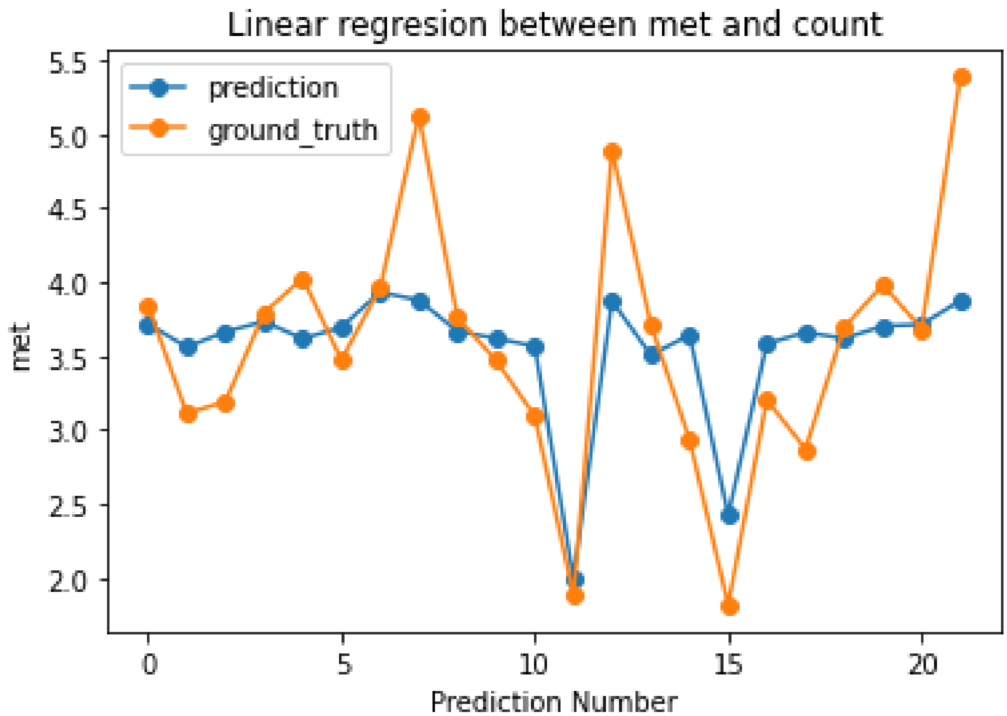
1 Model training

1.1 Linear regression model

[14]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

[15]:
```
cleaned_df = all_df.dropna(how='any')

x = cleaned_df['sum_mag_acc'].to_numpy().reshape(-1, 1)
y = cleaned_df['mean_met']

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)

lr = LinearRegression()
lr.fit(train_x,train_y)

pred_y = lr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Linear regresion between met and count')

print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(lr.score(train_x,train_y)))
```

```
mean squared error = 0.3445958198377466
r squared = 0.33717475205176983
```

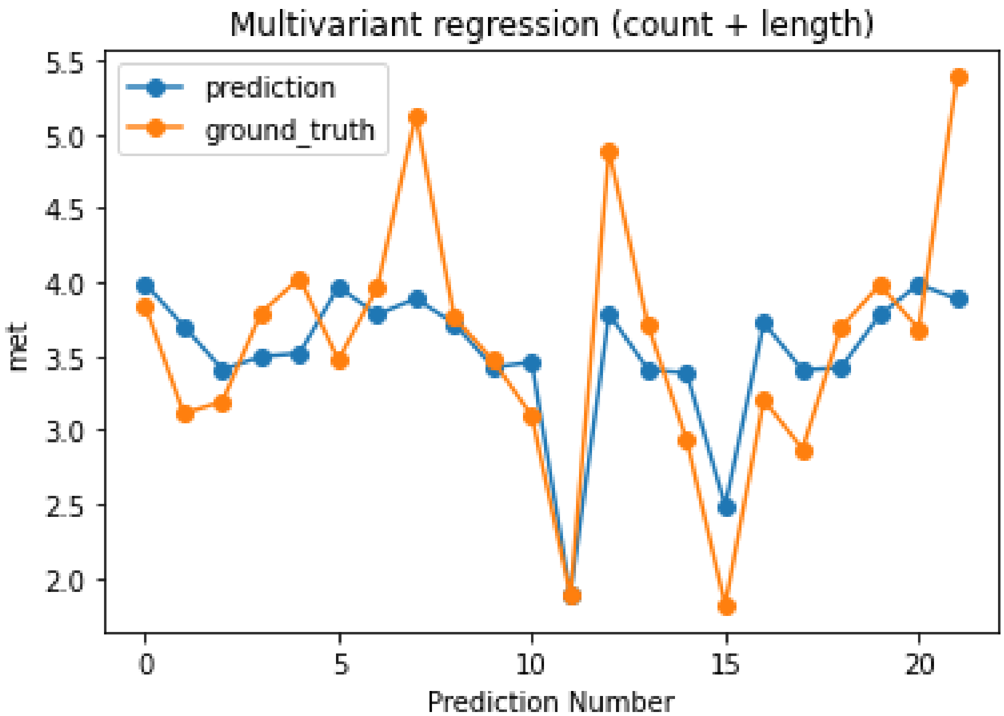


1.2 Multivariant regression model

1.2.1 Added length

[16]:
```
x = cleaned_df[['sum_mag_acc', 'length']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + length)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

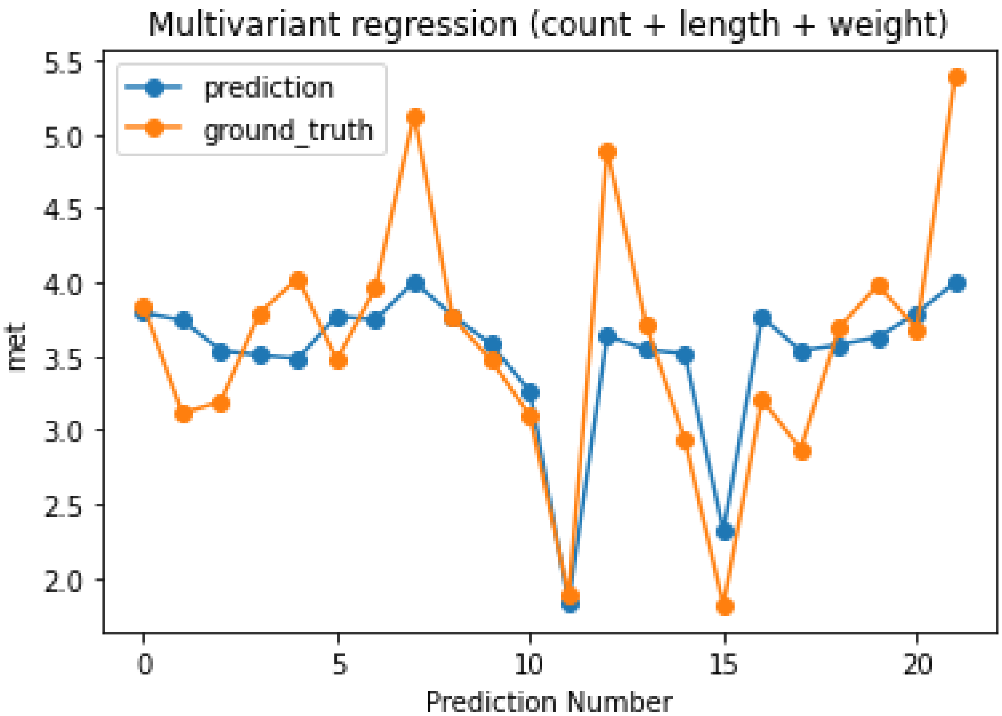
```
mean squared error = 0.3500677303401165
r squared = 0.38491558163531925
```



1.2.2 Added weight

[17]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + length +␣
 ↪weight)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

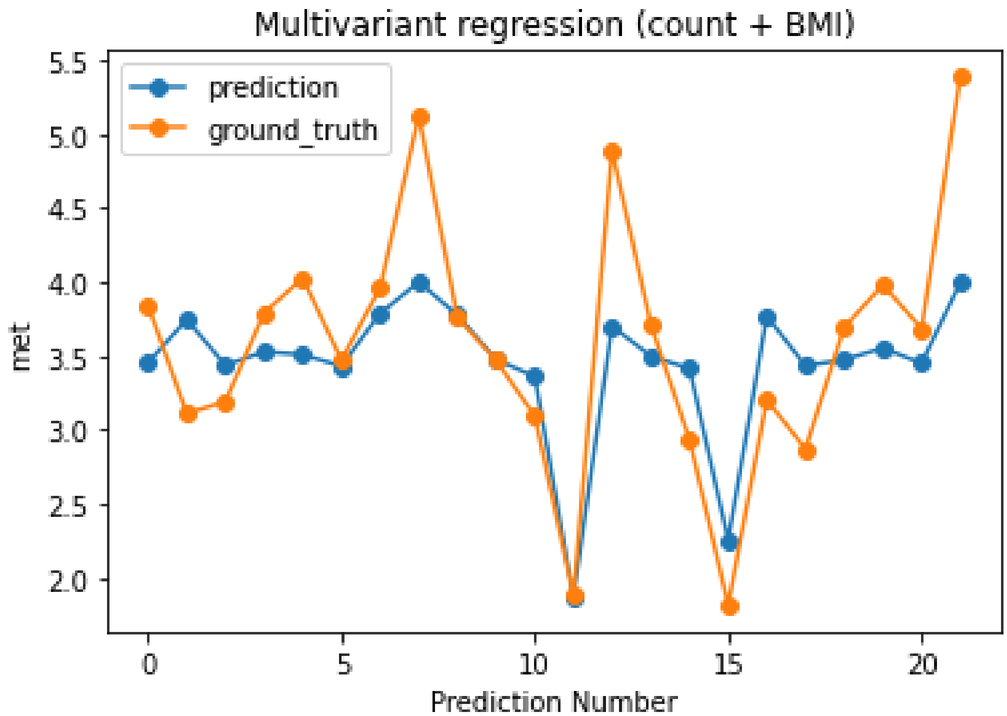
```
mean squared error = 0.33472358990356293
r squared = 0.4013578949855573
```



1.2.3 Added BMI with positive impact

[18]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

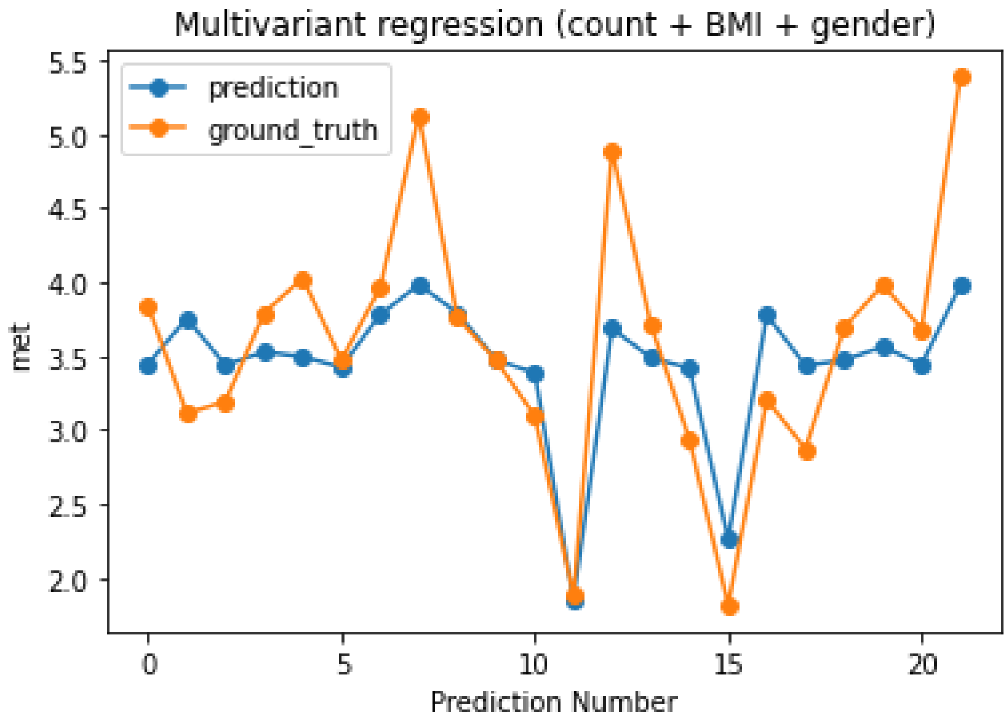
```
mean squared error = 0.3211026533049346
r squared = 0.4112314020476786
```



1.2.4 Added gender, with little positive impact

[19]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi', 'gender']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI +␣
 ↪gender)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

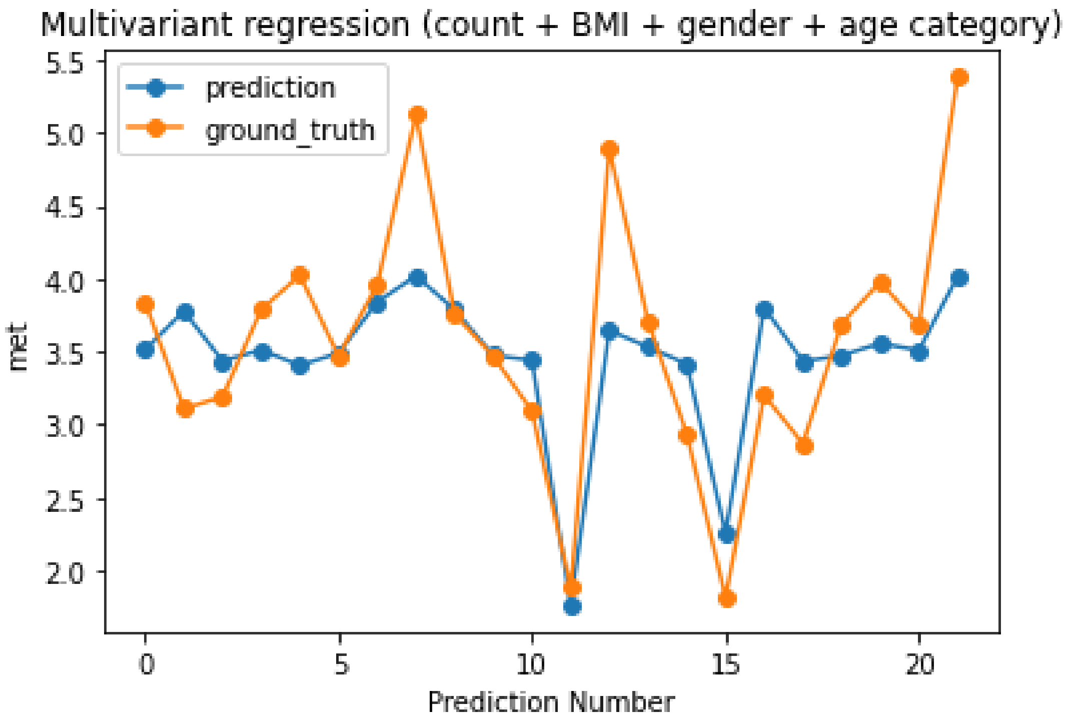
```
mean squared error = 0.33001082125020226
r squared = 0.411482822013164
```



1.2.5 Added age category, with positive impact

[20]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi', 'gender', 'age_cat']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI + gender␣
 ↪+ age category)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

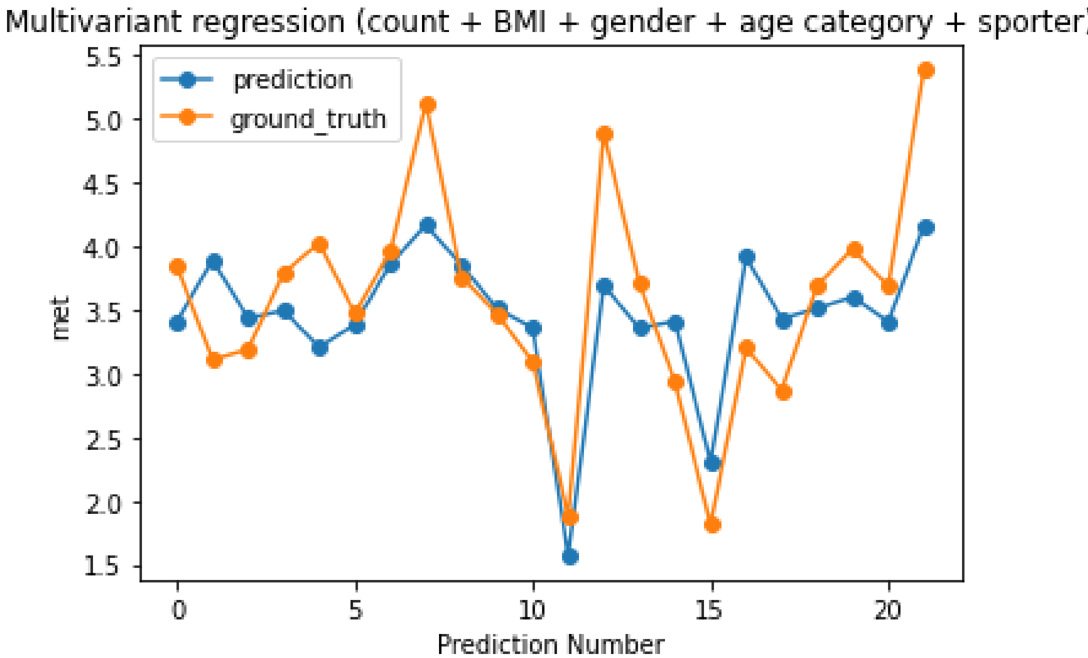
```
mean squared error = 0.3297091164105488
r squared = 0.41406699523606494
```



1.2.6 Added sporter, with positive impact

[21]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi', 'gender', 'age_cat',␣
↪'sporter']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI + gender␣
↪+ age category + sporter)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

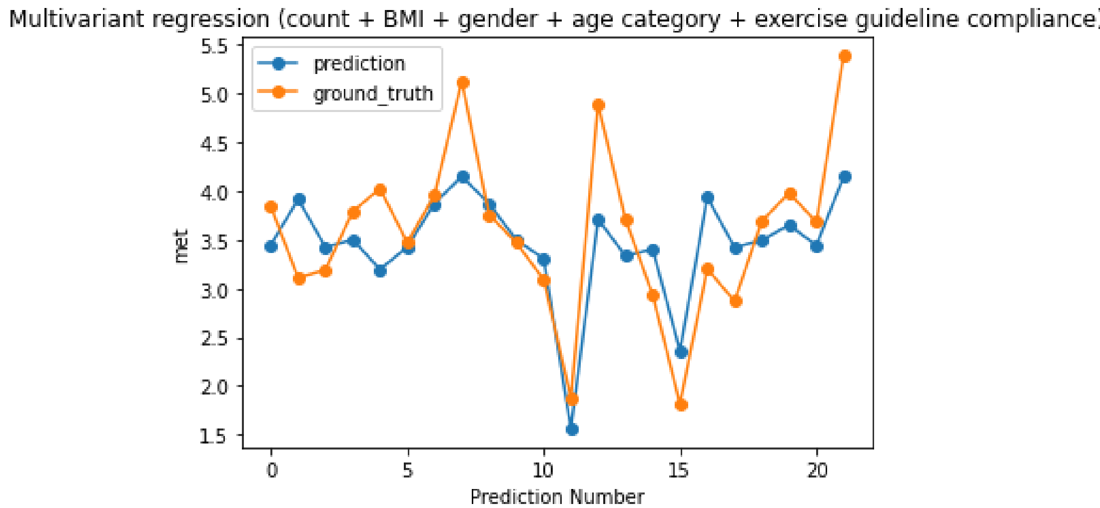
```
mean squared error = 0.3333184988883774
r squared = 0.4249970897901949
```



1.2.7 Added exercise guideline compliance

[22]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi', 'gender', 'age_cat',␣
↪'sporter', 'exercise_guideline_compliance']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI + gender␣
↪+ age category + exercise guideline compliance)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

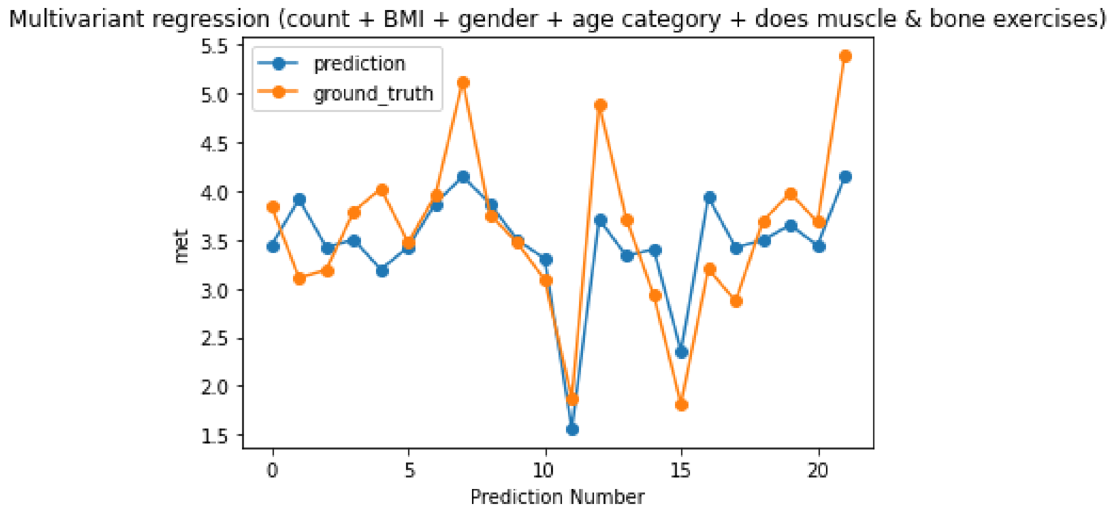
```
mean squared error = 0.33598145165666987
r squared = 0.42642390110842443
```



1.2.8 Added does muscle and bone exercises, with no impact

[23]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi', 'gender', 'age_cat',␣
↪'sporter', 'exercise_guideline_compliance', 'does_muscle_bone_exercises']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI + gender␣
↪+ age category + does muscle & bone exercises)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

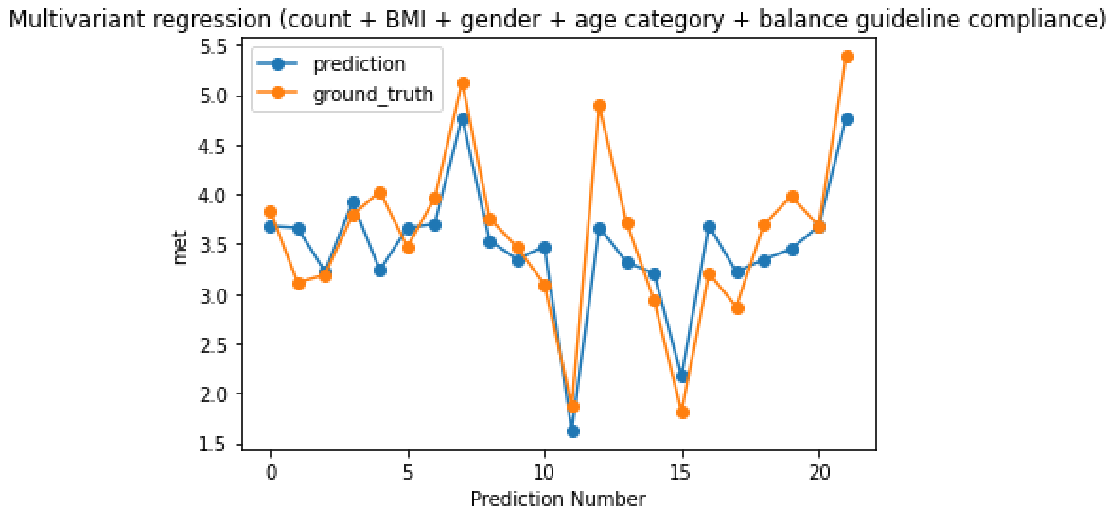
```
mean squared error = 0.3359814516566713
r squared = 0.4264239011084243
```



1.2.9 Added balance guideline compliance, with positive impact

[24]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi', 'gender', 'age_cat',␣
↪'sporter', 'exercise_guideline_compliance', 'balance_guideline_compliance']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI + gender␣
↪+ age category + balance guideline compliance)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

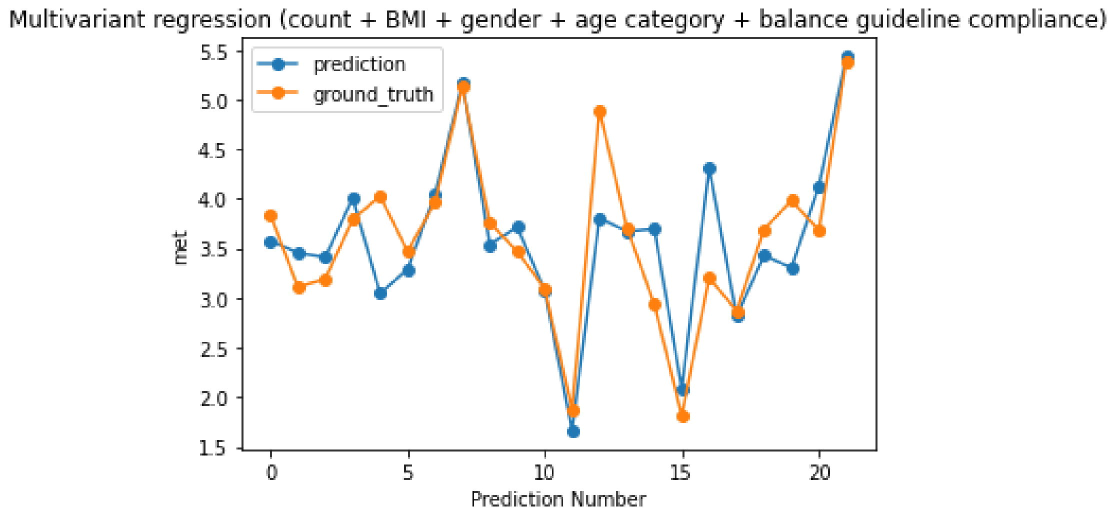
```
mean squared error = 0.20238917958311642
r squared = 0.496927650853599
```



1.2.10 Added speed, with positive impact

[25]:
```
x = cleaned_df[['sum_mag_acc', 'length', 'weight', 'bmi', 'gender', 'age_cat',␣
↪'sporter', 'exercise_guideline_compliance', 'balance_guideline_compliance',␣
↪'speed']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + BMI + gender␣
↪+ age category + balance guideline compliance)')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

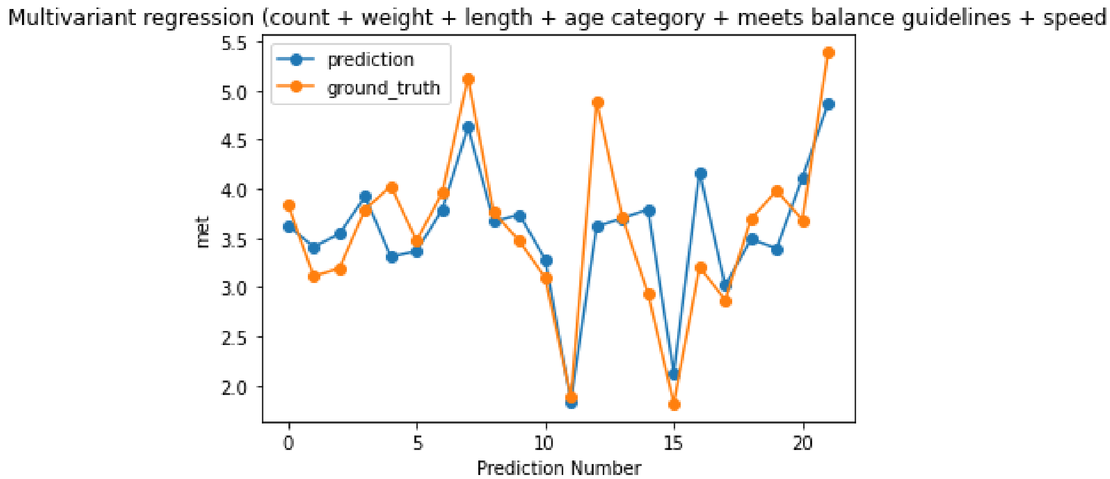
```
mean squared error = 0.23759231523113894
r squared = 0.6448383663078892
```



1.2.11 Multivariate regression best features

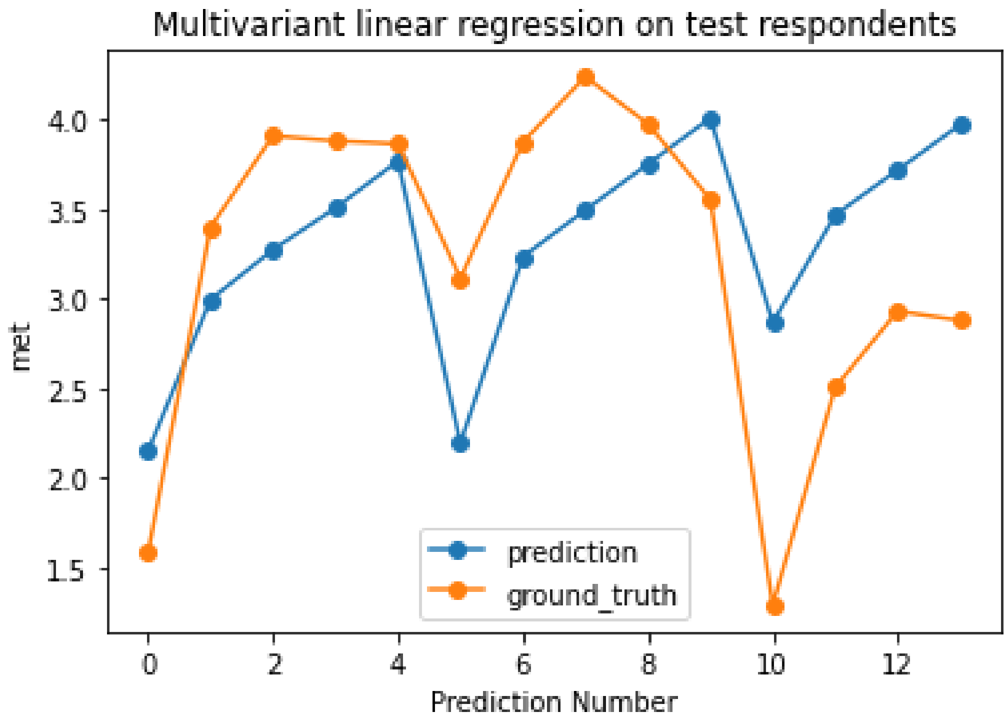
[26]:
```
x = cleaned_df[['sum_mag_acc', 'weight', 'length', 'age_cat',␣
↪'balance_guideline_compliance', 'speed']]
y = cleaned_df[['mean_met']]

train_x, valid_x, train_y, valid_y = train_test_split(x,y, test_size=0.2,␣
↪random_state=0)

mlr = LinearRegression()
mlr.fit(train_x,train_y)

pred_y = mlr.predict(valid_x)

plot_pred_truth(pred_y, valid_y, 'Multivariant regression (count + weight +␣
↪length + age category + meets balance guidelines + speed')
print('mean squared error = ' + str(mean_squared_error(valid_y, pred_y)))
print('r squared = ' + str(mlr.score(train_x,train_y)))
```

```
mean squared error = 0.24760849656566766
r squared = 0.5900683170230216
```



[27]:
```
test_x = test_df[['sum_mag_acc', 'weight', 'length', 'age_cat',␣
↪'balance_guideline_compliance', 'speed']]
test_y = test_df['mean_met']
pred_y = mlr.predict(test_x)

plot_pred_truth(pred_y, test_y, 'Multivariant linear regression on test␣
↪respondents')
print('mean squared error = ' + str(mean_squared_error(test_y, pred_y)))
print('r squared = ' + str(mlr.score(test_x,test_y)))
```

```
mean squared error = 0.5947081653355516
r squared = 0.21660042746074049
```



Multivariant linear regression on test respondents
met
4.0
3.5
3.0
2.5
2.0
1.5
0
2
4
6
8
10
12
Prediction Number
prediction
ground_truth


```
[ ]:
```