Report of Final Project

Predict the Type of Physical Activity

Group member:
Hefeng Lin hl3788
Haosong Zhang hz2244
Shengchao Sun ss12806
Hongyu Kang hk3068

In this project we are going to predict 4 different human activities (standing, walking, stairs down and stairs up) and using the given measures of these activities taken with an accelerometer of a smartphone.

The accelerometer measures the gravity acceleration that the device feels. The gravity acceleration in Earth is g=-9.8 m/s2. If one person is lying, he will feel an acceleration = g. A person going up in a rocket with an acceleration of 9.8 m/s2 will feel an acceleration a=2g. We feel similar situations each time when we go up and down in an elevator.

So, we can suppose that we could be able to predict different activities by studying the values of an accelerometer.

The dataset used for training in this project consists of two files. The first file is train_time_series.csv, contains the raw accelerometer data, which has been collected using the Beiwe research platform, and it has the following format:

```
In [8]: df_train_time_series
```

Out[8]:

	row_id	timestamp	UTC_time	accuracy	x_axis	y_axis	z_axis
0	20586	1565109930787	2019-08-06T16:45:30.787	unknown	-0.006485	-0.934860	-0.069046
1	20587	1565109930887	2019-08-06T16:45:30.887	unknown	-0.066467	-1.015442	0.089554
2	20588	1565109930987	2019-08-06T16:45:30.987	unknown	-0.043488	-1.021255	0.178467
3	20589	1565109931087	2019-08-06T16:45:31.087	unknown	-0.053802	-0.987701	0.068985
4	20590	1565109931188	2019-08-06T16:45:31.188	unknown	-0.054031	-1.003616	0.126450
...	...	...	...	...	...	...	...
3739	24325	1565110305638	2019-08-06T16:51:45.638	unknown	0.024384	-0.710709	0.030304
3740	24326	1565110305738	2019-08-06T16:51:45.738	unknown	0.487228	-1.099136	-0.015213
3741	24327	1565110305838	2019-08-06T16:51:45.838	unknown	0.369446	-0.968506	0.036713
3742	24328	1565110305939	2019-08-06T16:51:45.939	unknown	0.167877	-0.802826	0.049805
3743	24329	1565110306039	2019-08-06T16:51:46.039	unknown	0.689346	-0.991043	0.034973

3744 rows × 7 columns

In [12]:
```
df_train_labels
```

Out[12]:

	row_id	label
0	20589	1
1	20599	1
2	20609	1
3	20619	1
4	20629	1
...	...	...
370	24289	4
371	24299	4
372	24309	4
373	24319	4
374	24329	4

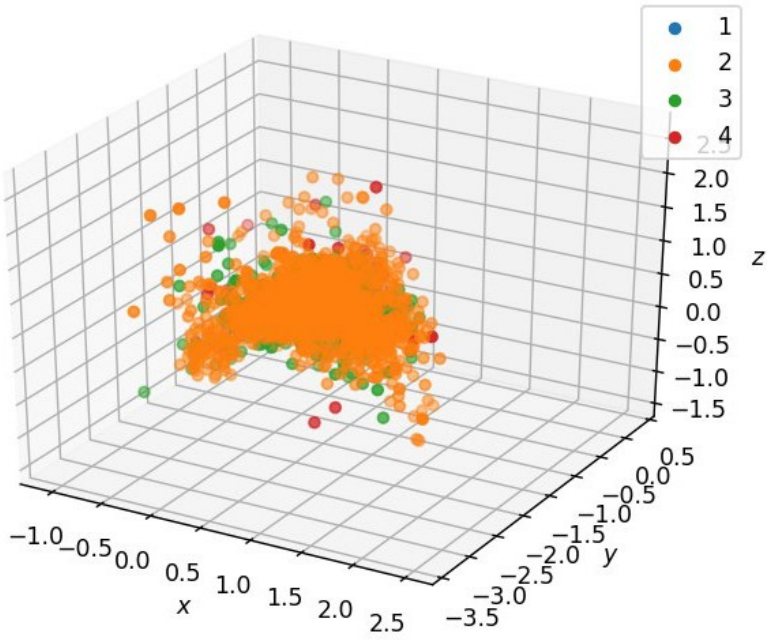
375 rows × 2 columns

The timestamp column is the time variable; and the last three columns, here labeled x, y, and z, which correspond to measurements of linear acceleration along each of the three orthogonal axes.

There is a second file its name is train_labels.csv, contains the activity labels, and we will using these labels to train the model. Different activities have been numbered with integers.

In the data file for training the model is include the labels. But we keep all the rows from the
file train_time_series.csv in order to have more data. The rows that don't have label, will have the same label as the last row that had one. Labels were informed originally in one of 10 rows. There are 1250 data in testing dataset, and 3744 data in training dataset.

To do an overview of the data, we first plot the three accelerations .

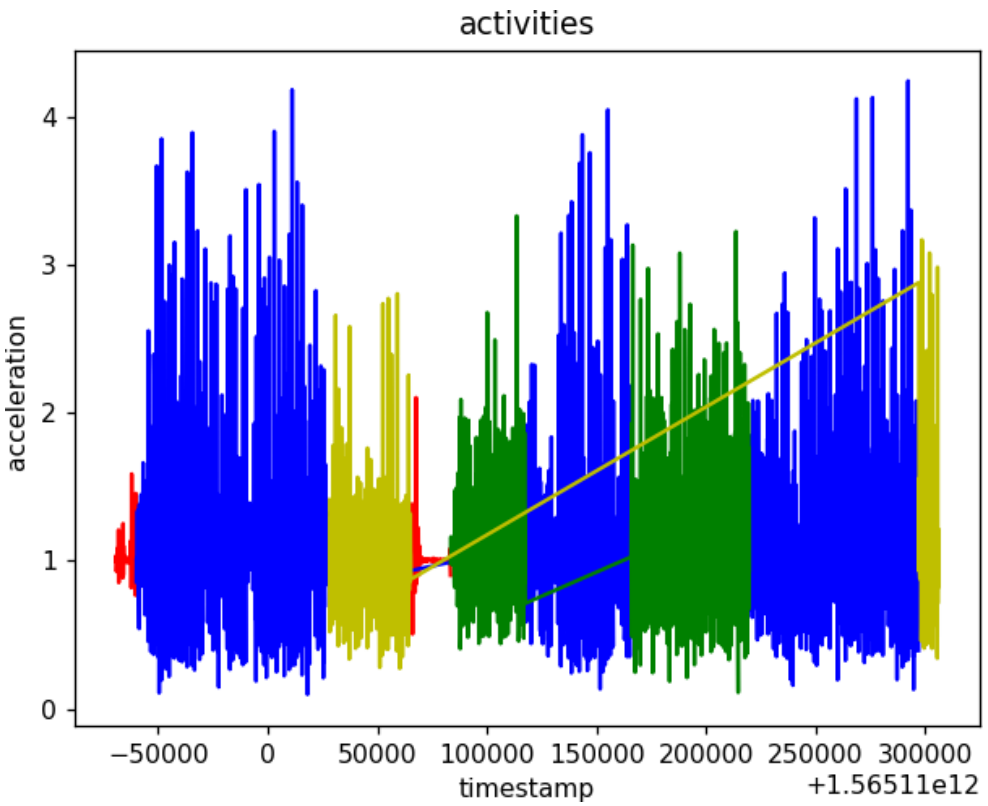


As we can see in the plot, there are many data from label 2 and the acceleration is not enough to classify the data.

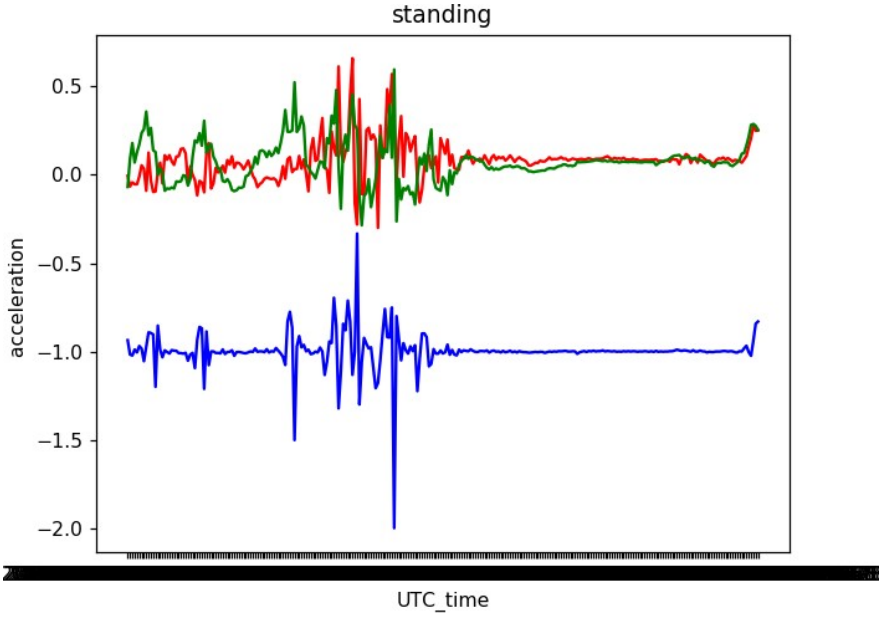
Our goal is to classify different physical activities as accurately as possible. To test our code, there is a file called test_time_series.csv, and at the end of the project we will provide the activity labels predicted by our code for this test data set. In both cases, for training and testing, the input file consists of a single (3-dimensional) time series. To test the accuracy of the code, we will upload the predictions as a CSV file. The file called test_labels.csv only contains the time stamps needed for prediction; we need to augment this file by adding the corresponding class predictions (1,2,3,4).

To take into account the three axes of the acceleration in just one variable, we calculate the
coefficient

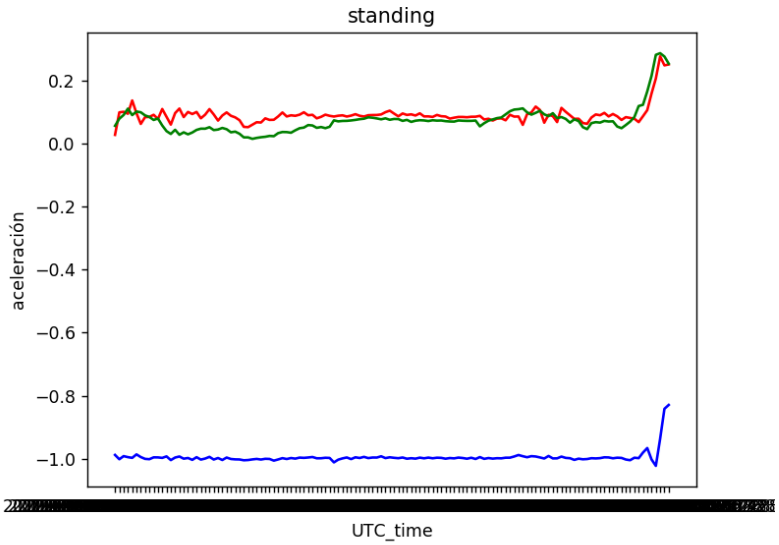
We plot time vs acceleration for each label to see if we need to clean it



For instance,
As the first half of the measure seems to be too noisy,



We are going to keep the second half of the data for the label 1 which represent standing



Then we do the same thing to label 2,3,4.
We're going to train one model taking into account the acceleration and the time. To do this, We are not going to use the coefficient of the acceleration with the three components because it doesn't add accuracy to the model. The components x and z are always around 0.

And to train the model and test it, we split the data like the picture

```
X_train, X_test, y_train,y_test=train_test_split(X,y,train_size=0.5,random_state=1)
```

Here we train the model looking for the coefficients that better fit the model. We try LogisticRegression because this is a classification model.

```
In [22]: from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(solver='liblinear', C=0.05,random_state=0,multi_class='ovr').fit(X_train, y_train)
print("Accuracy on training set is : {}".format(clf.score(X_train, y_train)))
print("Accuracy on test set is : {}".format(clf.score(X_test, y_test)))
y_test_pred = clf.predict(X_test)
print(classification_report(y_test, y_test_pred))
```

```
Accuracy on training set is : 0.32842105263157895
Accuracy on test set is : 0.28210526315789475
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        76
           2       0.00      0.00      0.00       126
           3       0.28      1.00      0.44       134
           4       0.00      0.00      0.00       139

    accuracy                           0.28       475
   macro avg       0.07      0.25      0.11       475
weighted avg       0.08      0.28      0.12       475
```

As we can see the accuracy is not very high and all the predictions will fall into label 3

When we consider periodic movements, like walking or climbing stairs, the movement has an assigned frequency. That's why a model that takes into account the frequency instead of the time, will be more accurate.

To study the system taking into account the frequency, we can translate our data to the frequency domain.

Instead of having the periodicity of the accelerations, we will have one value for each one of the elementary frequencies that compound the total movement.

To do this we can use the Fourier transform.

In [24]:
```
def fourier (df, freq, N):
    n=np.array(df).shape[0]
    x = fft(np.array(df))/n #(I can choose to normalize the transformation dividing between the number of rows)
    dt = 1 / (freq*N)#frequency is the inverse of the time and N is the number of elements in a period
    frq = fftfreq(n, dt)
#    return abs(x),frq
    return x.imag,frq

x1,freq1=fourier(df_train_1["y_axis"],0.1,80)
x2,freq2=fourier(df_train_2["y_axis"],0.1,13)
x3,freq3=fourier(df_train_3["y_axis"],0.1,96)
x4,freq4=fourier(df_train_4["y_axis"],0.1,90)
```

And we make our matrix with the variables and the vector with the output values of it. Then we split the data to have some data for training the model and some data to test it.

After constructing the matrices for the training and the test set, together with a list of the correct labels, we can use the scikit-learn package to construct a classifier. In this case we have use RandomForestClassifier.

In this case I will use RandomForestClassifier.

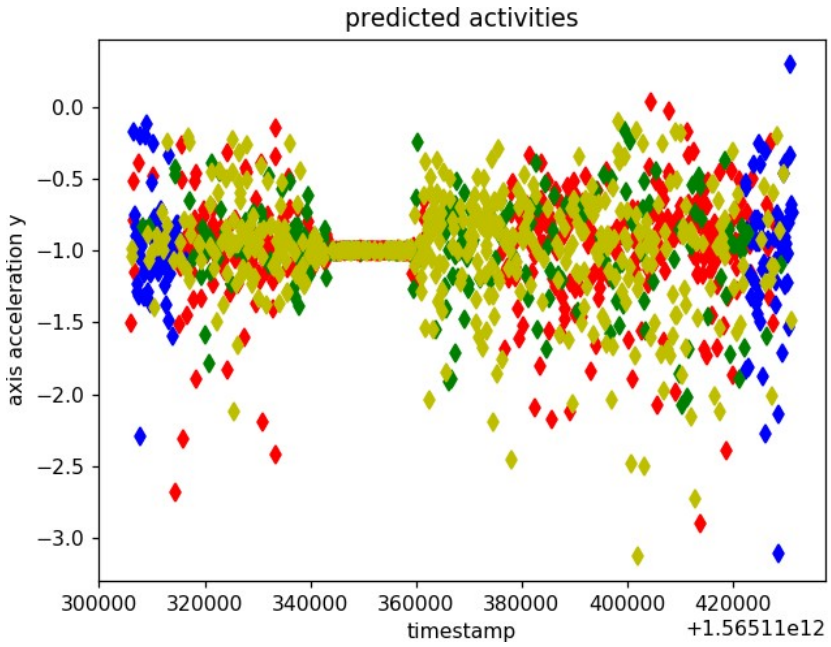
In [30]:
```
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
print("Accuracy on training set is : {}".format(clf.score(X_train, y_train)))
print("Accuracy on test set is : {}".format(clf.score(X_test, y_test)))
y_test_pred = clf.predict(X_test)
print(classification_report(y_test, y_test_pred))
```

```
Accuracy on training set is : 0.9978947368421053
Accuracy on test set is : 0.6063157894736843
              precision    recall  f1-score   support

           1       0.76      0.70      0.73        76
           2       0.74      0.83      0.78       126
           3       0.47      0.51      0.49       134
           4       0.52      0.45      0.48       139

    accuracy                           0.61       475
   macro avg       0.62      0.62      0.62       475
weighted avg       0.60      0.61      0.60       475
```

Here I'm going to predict the activities from the data set test, that we renamed as questions, And I'm going to plot the results



Conclusion:

The first model that we tried, LogisticRegression in the time dimensions for the acceleration in y axis has a very low accuracy on the training set and on the test set. The only label that has predictions is label 3.

	precision	recall	f1-score	support
1	0.00	0.00	0.00	76
2	0.00	0.00	0.00	126
3	0.28	1.00	0.44	134
4	0.00	0.00	0.00	139
accuracy			0.28	475

After cleaning the data, the accuracy was 0.28, so to clean the data improved a little the predictions, but not much. And in the plot of the test data, we can see as all the rows fall into label 3.

As evident just looking at the plot that not all the dataset is stairs up. There is a line of standing.

As all the activities (unless standing) are periodic movements, we thought it would be a good idea to study the activities looking at their periodicity. So we did some research and find the Fourier Transform is a tool that converts periodic movements in the time domain to its frequencies in the frequency domain.

When we plot the Fourier Tranforms, we can see the frequencies that compose each movement, but maybe it's too noisy. We see a lot of frequencies and we guess the model is not accurate enough to classify the activities with so many frequencies. We expected to have one main frequency and some few little more (2 or 3.)

	precision	recall	f1-score	support
1	0.76	0.72	0.74	76
2	0.73	0.83	0.78	126
3	0.48	0.51	0.49	134
4	0.53	0.44	0.48	139
accuracy			0.61	475

Also it is possible to drop the frequencies with lower significance.

This is what we obtained from a RandomForestClassification in the frequency domain for the acceleration in y axis.