Cpts_415: Big Data

Project Final Report

Project name: Activity Analyzer with Wearable Sensors

12/08/2019

Team members:

1. XinYi Wang (11653817)
2. Yu-Chieh Wang (11641327)
3. Thunradee Tangsupakij (11616232)

Instructor: Tinghui Wang

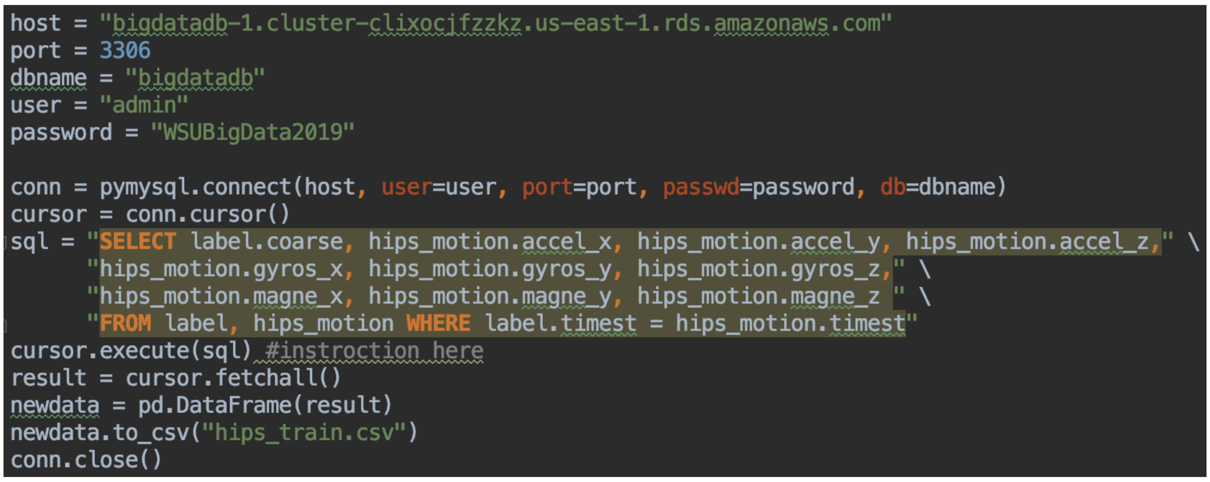
1. Introduction: Observe data set from wearable sensors of Sussex-Huawei Locomotion, make visualization to show the relationship between the data, and use a machine learning model to analyze and predict activities. We want to focus on people's motion patterns. We want to figure out how much time that people use to transport can be accounted for by different traffic tools. With the technology of wearable sensors today, we can monitor people's motion patterns to decide which traffic tools they are using. Our team is working on making a machine-learning algorithm to construct a model and use it to predict activities.

   Motivation: We are willing to analyze people's motion patterns over a period of time and trying to give scientific advice through resulting in the data analysis. For example, if some people lack physical exercise then the device is able to send a notification to the users and it may encourage the users to make changes toward a healthy lifestyle.

Problem definition: Decision for using which kind of algorithm is the main problem for us. We need to figure out a model that can fit the goad and training data as possible as we can.

2. Related Work:

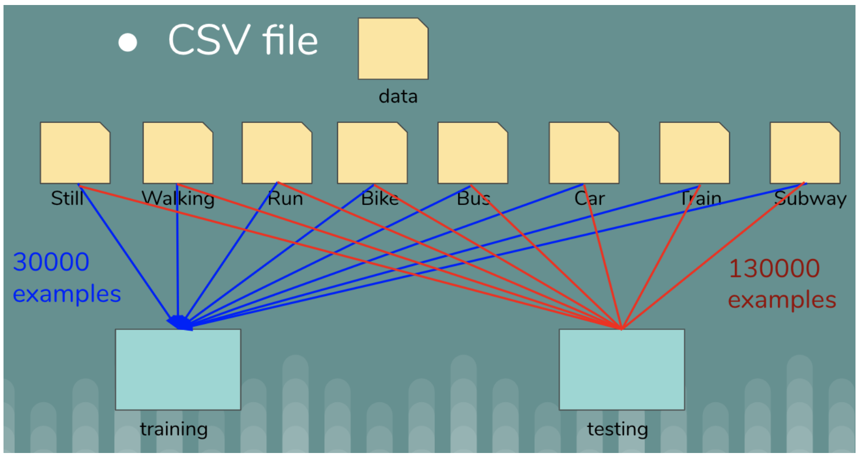
   Since we thought it took too much time to get the data from the database, we decided to download all the data and analyze it with a program we wrote. The next steps are our experiments. First of all, we write a Python code to access our database by SQL, download the data we need, and make it as a CSV file.



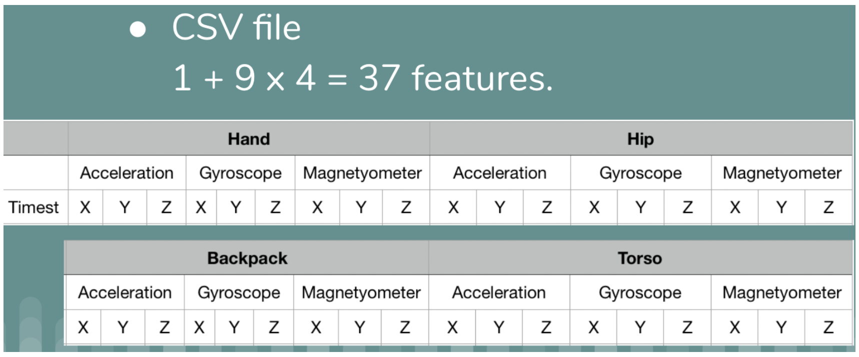
```
host = "bigdatadb-1.cluster-clixocjfzzkz.us-east-1.rds.amazonaws.com"
port = 3306
dbname = "bigdatadb"
user = "admin"
password = "WSUBigData2019"

conn = pymysql.connect(host, user=user, port=port, passwd=password, db=dbname)
cursor = conn.cursor()
sql = "SELECT label.coarse, hips_motion.accel_x, hips_motion.accel_y, hips_motion.accel_z," \
      "hips_motion.gyros_x, hips_motion.gyros_y, hips_motion.gyros_z," \
      "hips_motion.magne_x, hips_motion.magne_y, hips_motion.magne_z " \
      "FROM label, hips_motion WHERE label.timest = hips_motion.timest"
cursor.execute(sql) #instroction here
result = cursor.fetchall()
newdata = pd.DataFrame(result)
newdata.to_csv("hips_train.csv")
conn.close()
```

Second, we separate this CSV file to 8 CSV files by 8 activities so that we can pick some examples as our training data and use the rest of the examples as our testing data. Third, we random the order of the examples of our training and testing data. Next, We added the training data to the decision number algorithm to obtain a training model, and then put the test data into the training model to obtain the prediction result. Finally, the prediction result is compared with the actual label to get the prediction accuracy.



3. Dataset forming: We extract pure data from the original dataset to form a pure dataset for each activity. Then we take 5 mins data(30000 examples pre-activity) as our training data and the rest(130000 examples pre-activity) as the testing data, depending on the frequency at which the sensor collects the data. For the missing data and null data, we just directly drop them off from the dataset in case it has a bad influence on the pure dataset. The datasets we use for training and testing are consist of 37 labels which are label and 36 features from 4 sensors.



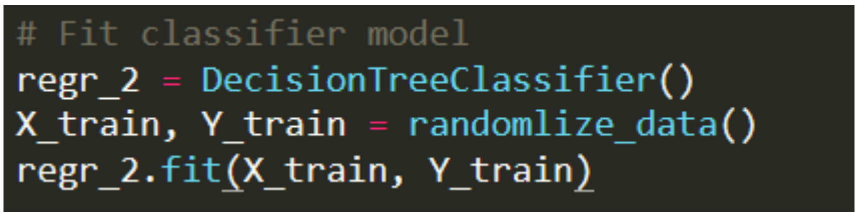
- CSV file
  1 + 9 x 4 = 37 features.

	Hand									**Hip**								
	Acceleration			Gyroscope			Magnetyometer			Acceleration			Gyroscope			Magnetyometer		
Timest	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z

Backpack									**Torso**								
Acceleration			Gyroscope			Magnetyometer			Acceleration			Gyroscope			Magnetyometer		
X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z

Model: First, we remodeling the raw data into the format that we want by using a Python data processing tool. Then we randomize the data but keep the label-features relation fixed. Finally, we put the features set and label set into the algorithm to fit it. In order to see the output result for the prediction accuracy, we use a specific command line from the scikit-learn decision tree which called accuracy_score. We can also visualize the instances that are correctly classified and not correctly classified through printing out the confusion matrix.

Algorithm: We decide to use the DecisionTreeClassifier algorithm from the library of scikit-learn.

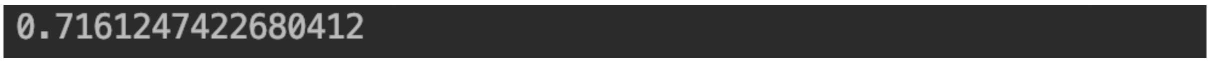


```
# Fit classifier model
regr_2 = DecisionTreeClassifier()
X_train, Y_train = randomlize_data()
regr_2.fit(X_train, Y_train)
```

Method: Here we want to point out some important methods that may impact the final result. First is the way we manage the raw data. We pick the data for individual activity from the pure data segment, which means we pick one-day data and filter out the target label we want with the features along with so that we can make sure the data we picked is especially for specific one activity. Secondly, it includes a mechanism inside of the algorithm that converts the continuous values to discrete values. The true method for it is, the algorithm finds the maximal and minimal value for one feature as an interval, then divides it into several small intervals with same space between, so that the algorithm can automatically classify multiple continuous values into several small labels(here the # of small labels are equal to the # of small intervals with same space). For example, if the maximal and minimal value is +10 and -10, then the interval can be [-10, +10], and if it can be divided into 2 intervals then it will be [-10, 0] and [0, +10]. If one of the values is 5 in one feature value set, it will be classified as label 1. If one of the values is -2 in one feature value set, it will be classified as label 0.

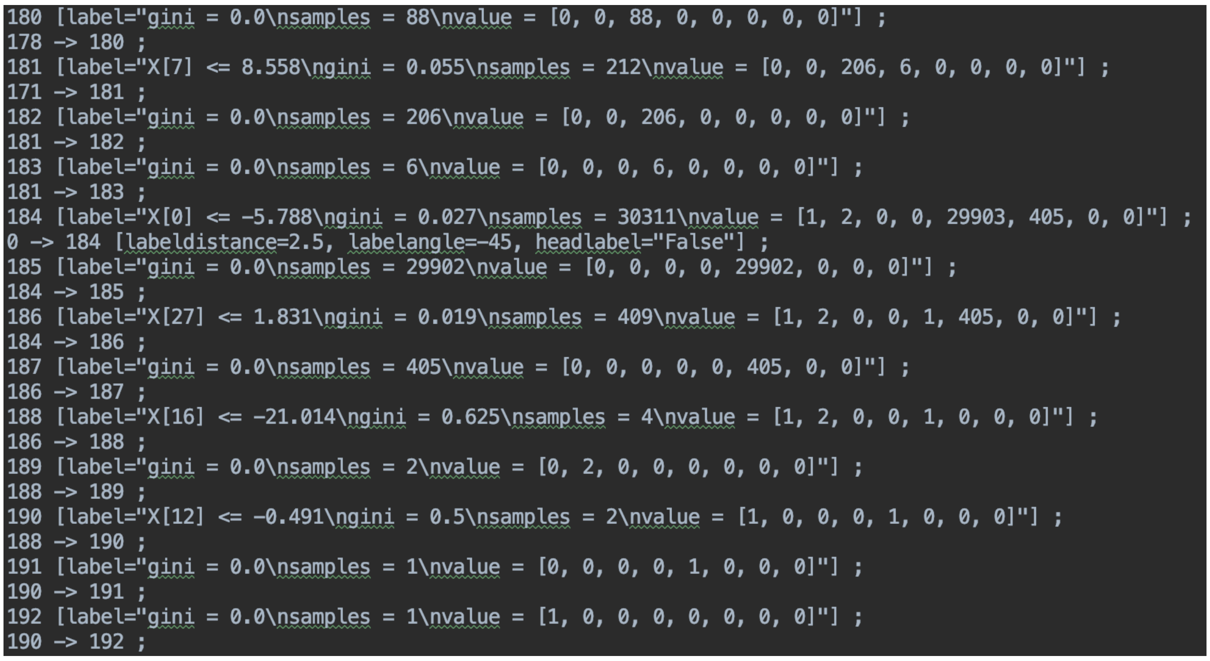
4. Results and findings:

   The result shows that the decision tree algorithm works and gets the accuracy which is up to 71.61%.



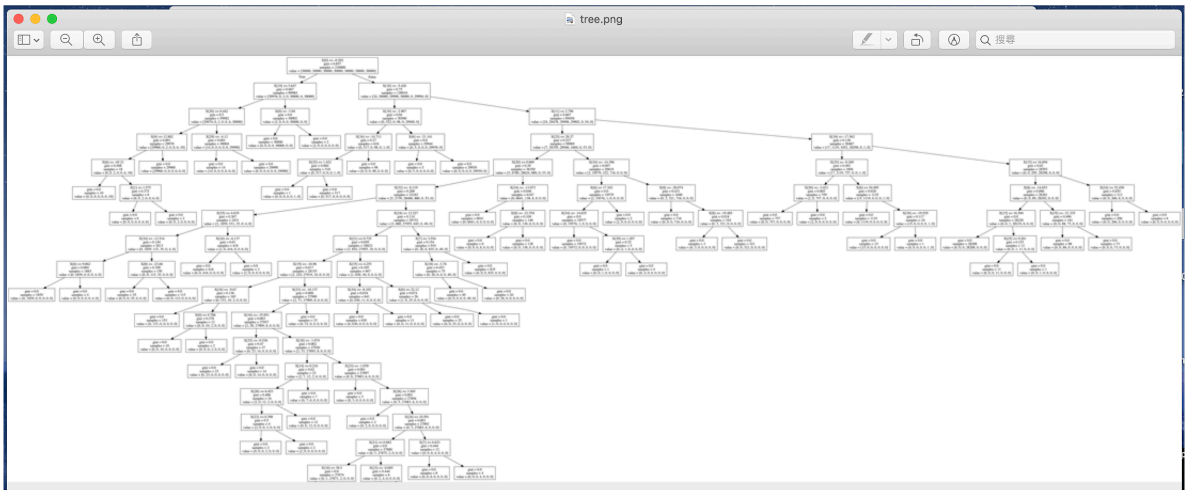
```
0.7161247422680412
```

The following picture show there are a total of 190 dots in our decision tree model.

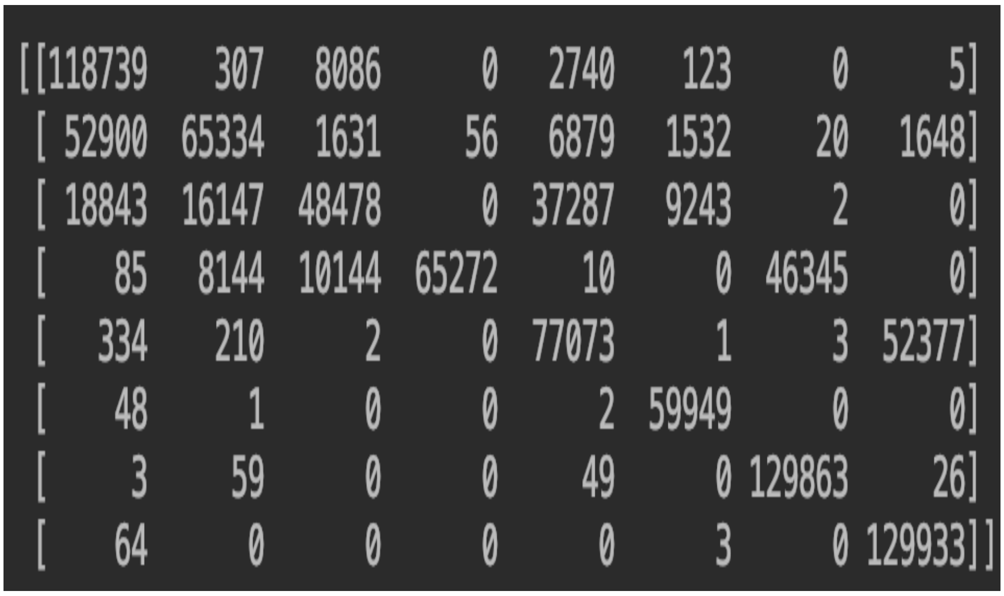


```
180 [label="gini = 0.0\nsamples = 88\nvalue = [0, 0, 88, 0, 0, 0, 0, 0]"] ;
178 -> 180 ;
181 [label="X[7] <= 8.558\ngini = 0.055\nsamples = 212\nvalue = [0, 0, 206, 6, 0, 0, 0, 0]"] ;
171 -> 181 ;
182 [label="gini = 0.0\nsamples = 206\nvalue = [0, 0, 206, 0, 0, 0, 0, 0]"] ;
181 -> 182 ;
183 [label="gini = 0.0\nsamples = 6\nvalue = [0, 0, 0, 6, 0, 0, 0, 0]"] ;
181 -> 183 ;
184 [label="X[0] <= -5.788\ngini = 0.027\nsamples = 30311\nvalue = [1, 2, 0, 0, 29903, 405, 0, 0]"] ;
0 -> 184 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
185 [label="gini = 0.0\nsamples = 29902\nvalue = [0, 0, 0, 0, 29902, 0, 0, 0]"] ;
184 -> 185 ;
186 [label="X[27] <= 1.831\ngini = 0.019\nsamples = 409\nvalue = [1, 2, 0, 0, 1, 405, 0, 0]"] ;
184 -> 186 ;
187 [label="gini = 0.0\nsamples = 405\nvalue = [0, 0, 0, 0, 0, 405, 0, 0]"] ;
186 -> 187 ;
188 [label="X[16] <= -21.014\ngini = 0.625\nsamples = 4\nvalue = [1, 2, 0, 0, 1, 0, 0, 0]"] ;
186 -> 188 ;
189 [label="gini = 0.0\nsamples = 2\nvalue = [0, 2, 0, 0, 0, 0, 0, 0]"] ;
188 -> 189 ;
190 [label="X[12] <= -0.491\ngini = 0.5\nsamples = 2\nvalue = [1, 0, 0, 0, 1, 0, 0, 0]"] ;
188 -> 190 ;
191 [label="gini = 0.0\nsamples = 1\nvalue = [0, 0, 0, 0, 1, 0, 0, 0]"] ;
190 -> 191 ;
192 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0, 0, 0]"] ;
190 -> 192 ;
```

Then, we draw these points as an entity decision tree as following:



Finally, we make a confusion matrix so that we can see which label has higher accuracy and which label is hard to be classified.

Still	
Walking	
Run	
Bike	
Car	
Bus	
Train	
Subway	

True/ Prediction	Still	Walking	Run	Bike	Car	Bus	Train	Subway

```
[[118739    307   8086      0   2740    123      0      5]
 [ 52900  65334   1631     56   6879   1532     20   1648]
 [ 18843  16147  48478      0  37287   9243      2      0]
 [    85   8144  10144  65272     10      0  46345      0]
 [   334    210      2      0  77073      1      3  52377]
 [    48      1      0      0      2  59949      0      0]
 [     3     59      0      0     49      0 129863     26]
 [    64      0      0      0      0      3      0 129933]]
```

According to the plot, we can see that each activity(row) has 130000 examples. In addition, we find some interesting things in this plot. First of all, Subway has the highest accuracy which means people shaking on the subway are very consistent. Second, when people walk, they don’t walk all the time, but stop and go. As a result, we can see some activity examples are misclassified, but some of the activities have high accuracies.

Contribution in the group:

XinYi Wang: Design prediction model and implement in Python language, plot the result and do with the output analysis. Complete all the milestones, final report and presentation slides.

Yu-Chieh Wang: Analysis of the topic and design the method of processing the dataset. Implement of extracting pure data of each activity from the original database and reshape the dataset chose to the data we need. Do an analysis of the dataset. Complete all the milestones, final report and presentation slides.

Thunradee Tangsupakij: Analysis of the topic and design the method of processing the dataset. Related work for the project and provides multiple methods to form the database we might use. Complete all the milestones, final report and presentation slides.