ESWEEK IoMT Design Contest Project Report

Embedded Human Activity Recognition Design Contest

Burak Albayrak

Istanbul Sehir University

Contents

Purpose of the Project

As any other living being on this planet humans depend also on usage of their muscles. use different muscles in our body for different actions ranging from basic activities to hard workouts. The way things are going in the industry we might be inclined to ask that what can we do to maximize our muscle usage in terms of efficiency using the technologies available? The answer is wearable devices. Wearable devices can give us the opportunity to collect information about how muscles are working. These devices can also be used in human activity recognition. Human activity recognition makes it possible to use information obtained by subjects for specific purposes in certain wearable devices. Generally, these can be count as following; to make disabled people's lives easier, getting information from athletes and according to this information, setting-up daily workout training for individual athletes, to prevent possible injuries during sports activities and to minimize the recovery time of a patient receiving physiotherapeutic treatment.

In this project, we used wearable devices as a monitor of user activity. The purpose of the project is that recognize and define the current condition of the subject according to human activities. Subjects' information is stored in CSV files. These CSV files keep data of each subject’s “Time(s)”, “User”, “Scenario”, “Trial”, “Window Number”, “Stretch Value” and “Label” variables. Time(s) variable is showing milliseconds, Scenario is combinations of sequence of activities and Label is for current activity. With the provided data we started to turn the data to a data frame with the use of Jupiter notebook.

Definitions

A Decision Tree is a structure used to divide a data set containing a large number of records into smaller clusters by applying a set of decision rules. In summary it is a structure used by dividing large amounts of records into very small groups of records by applying simple decision-making steps.

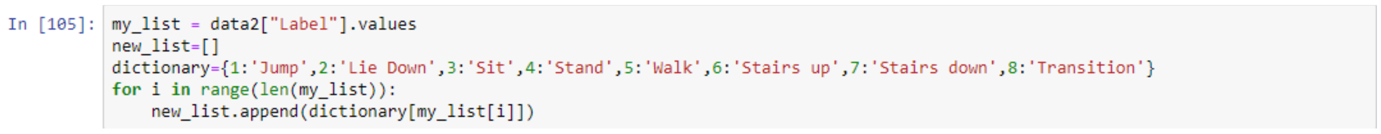
Random Forest is result of multiple decision trees. During the classification process, it is aimed to increase the classification value by using more than one decision tree.

Tools

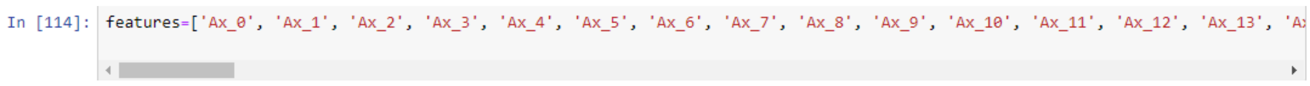
We have done classification on Jupyter Notebook with the help of the Python 3.7 language. Simplicities that provided to us by Jupyter notebook, we have turned the user data files from CSV format to Panda data frame. For several mathematical calculations, numpy was used. Graphs are made by matplotlib,seaborn, graphviz, pydotplus tools. Train and test data sets while creating Decision Tree are created with sklearn module.

Procedure

Files were in csv format. With Jupiter notebook which uses python, I turned CSV files into the data frames.

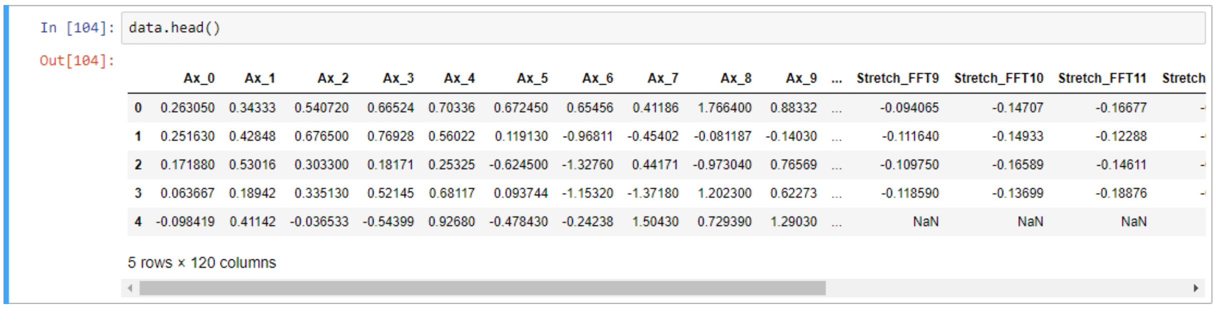


```
In [105]: my_list = data2["Label"].values
          new_list=[]
          dictionary={1:'Jump',2:'Lie Down',3:'Sit',4:'Stand',5:'Walk',6:'Stairs up',7:'Stairs down',8:'Transition'}
          for i in range(len(my_list)):
              new_list.append(dictionary[my_list[i]])
```



```
In [114]: features=['Ax_0', 'Ax_1', 'Ax_2', 'Ax_3', 'Ax_4', 'Ax_5', 'Ax_6', 'Ax_7', 'Ax_8', 'Ax_9', 'Ax_10', 'Ax_11', 'Ax_12', 'Ax_13', 'A
```

Decision Trees classifies the data according to the parameters in the list of features shown above and divides the data into the smallest units. In this way, we receive a summary of the data. That features list contains 120 values. These values keeps angle between muscles, stretch values, minimum stretch, maximum stretch data.

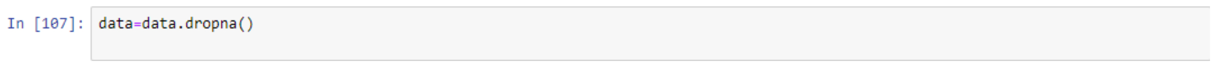


```
In [104]: data.head()
Out[104]:
```

	Ax_0	Ax_1	Ax_2	Ax_3	Ax_4	Ax_5	Ax_6	Ax_7	Ax_8	Ax_9	...	Stretch_FFT9	Stretch_FFT10	Stretch_FFT11	Stretch
0	0.263050	0.34333	0.540720	0.66524	0.70336	0.672450	0.65456	0.41186	1.766400	0.88332	...	-0.094065	-0.14707	-0.16677	-
1	0.251630	0.42848	0.676500	0.76928	0.56022	0.119130	-0.96811	-0.45402	-0.081187	-0.14030	...	-0.111640	-0.14933	-0.12288	-
2	0.171880	0.53016	0.303300	0.18171	0.25325	-0.624500	-1.32760	0.44171	-0.973040	0.76569	...	-0.109750	-0.16589	-0.14611	-
3	0.063667	0.18942	0.335130	0.52145	0.68117	0.093744	-1.15320	-1.37180	1.202300	0.62273	...	-0.118590	-0.13699	-0.18876	-
4	-0.098419	0.41142	-0.036533	-0.54399	0.92680	-0.478430	-0.24238	1.50430	0.729390	1.29030	...	NaN	NaN	NaN	

5 rows × 120 columns

As seen we have NAN values which means empty for that specific column and row. To use dataframe modules accurately we have to drop these rows from the dataframe.

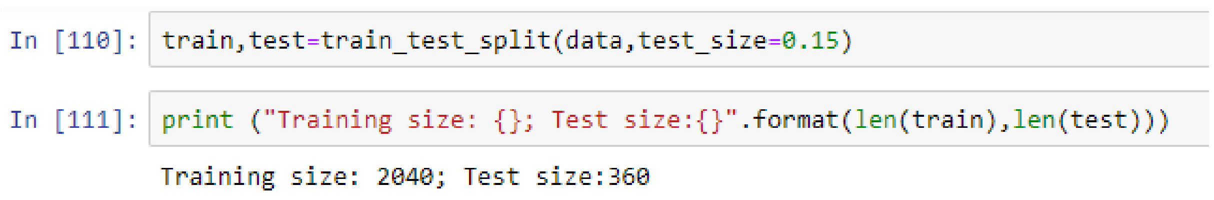


```
In [107]: data=data.dropna()
```

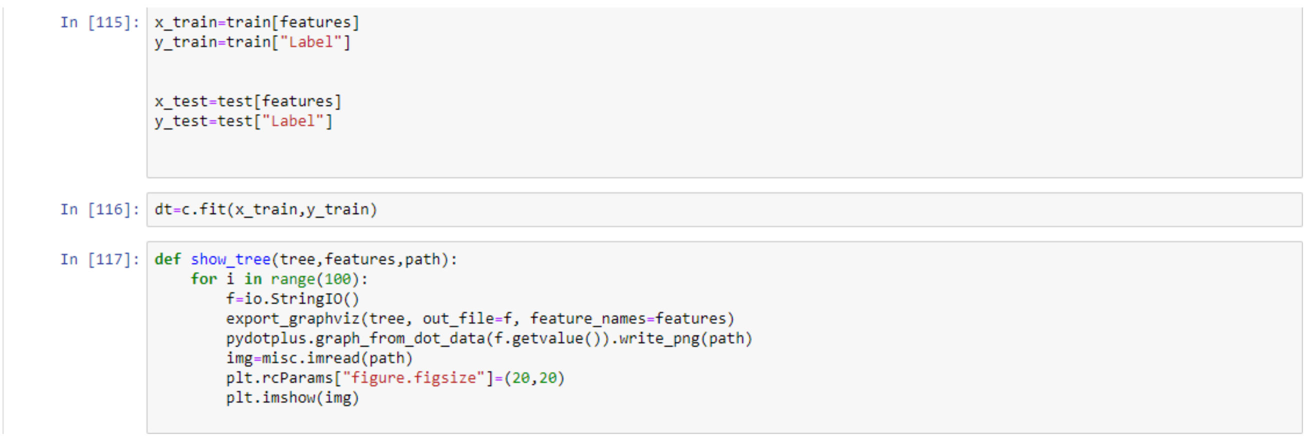
With that piece of code we easily can get rid off those unwanted rows.

The Program and Results

After we get clear rows then we can move on with training and testing. At last we will get accuracy of the predictions of different classifiers.

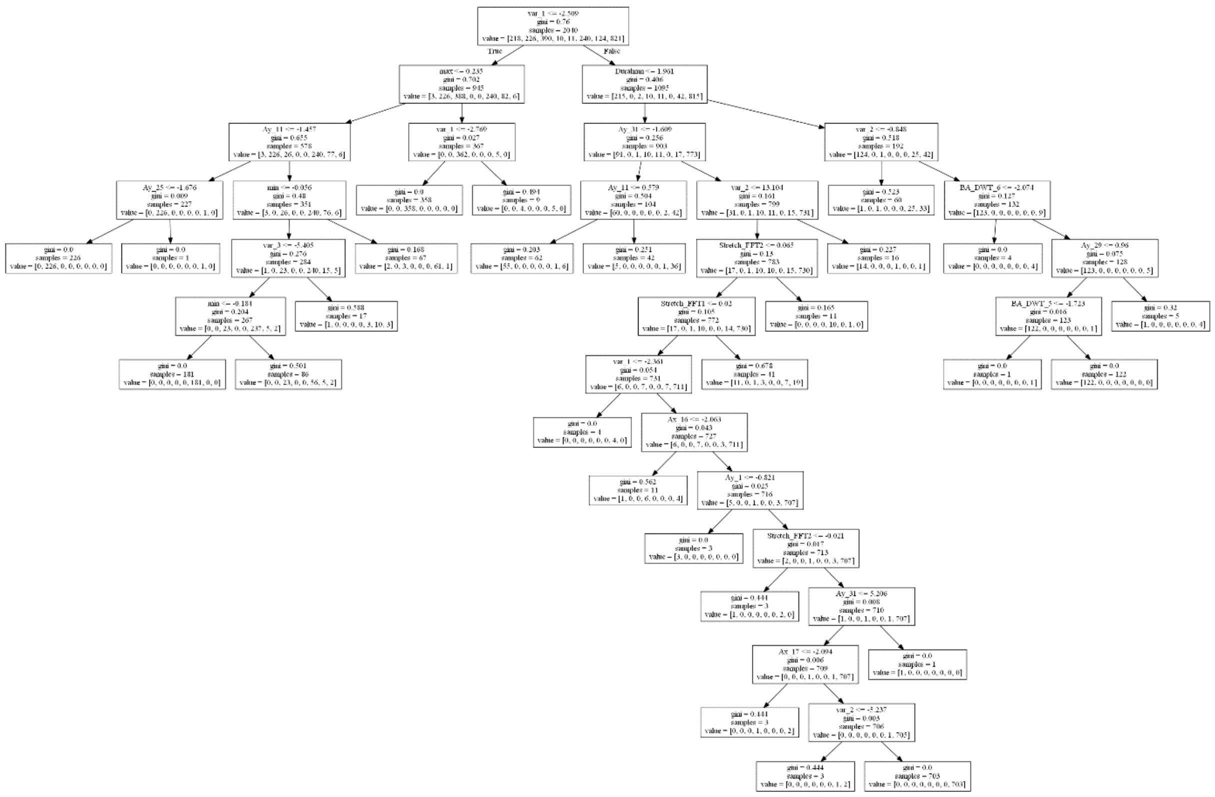


```
In [110]: train,test=train_test_split(data,test_size=0.15)

In [111]: print ("Training size: {}; Test size:{}".format(len(train),len(test)))

          Training size: 2040; Test size:360
```



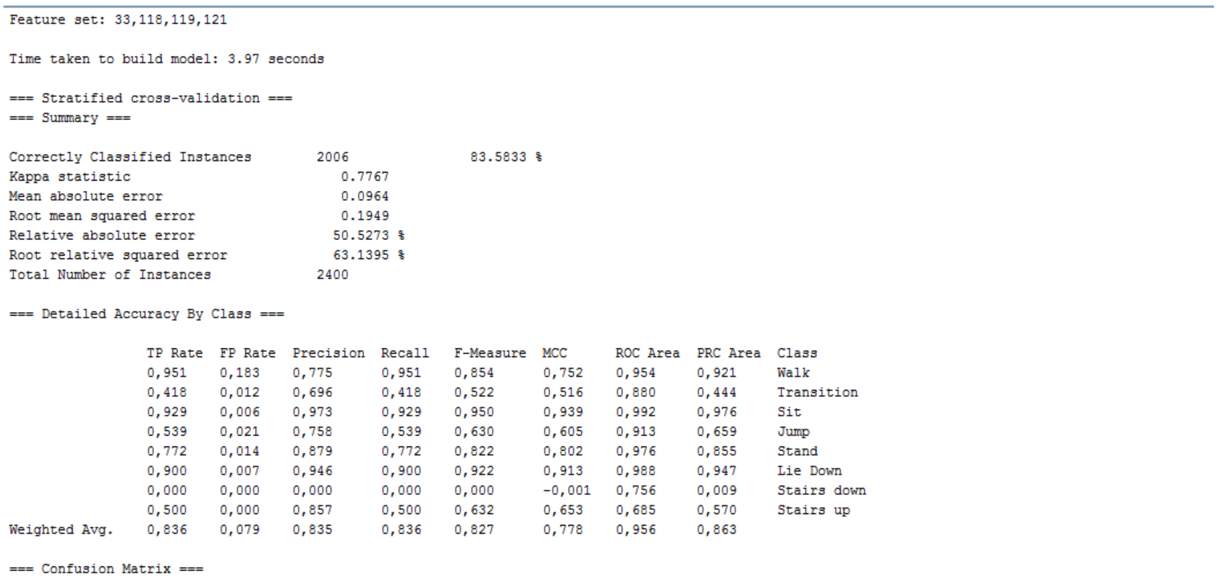
```
In [115]: x_train=train[features]
          y_train=train["Label"]


          x_test=test[features]
          y_test=test["Label"]


In [116]: dt=c.fit(x_train,y_train)

In [117]: def show_tree(tree,features,path):
              for i in range(100):
                  f=io.StringIO()
                  export_graphviz(tree, out_file=f, feature_names=features)
                  pydotplus.graph_from_dot_data(f.getvalue()).write_png(path)
                  img=misc.imread(path)
                  plt.rcParams["figure.figsize"]=(20,20)
                  plt.imshow(img)
```

And after all these we get decision trees as shown below



var_1 <= -2.509 gini = 0.76 samples = 2040 value = [218, 226, 390, 10, 11, 240, 124, 821]
True
False
max <= 0.235 gini = 0.702 samples = 945 value = [3, 226, 388, 0, 0, 240, 82, 6]
Duration <= 1.961 gini = 0.406 samples = 1095 value = [215, 0, 2, 10, 11, 0, 42, 815]
Ay_11 <= -1.457 gini = 0.655 samples = 578 value = [3, 226, 26, 0, 0, 240, 77, 6]
var_1 <= -2.769 gini = 0.027 samples = 367 value = [0, 0, 362, 0, 0, 0, 5, 0]
Ay_31 <= -1.609 gini = 0.256 samples = 903 value = [91, 0, 1, 10, 11, 0, 17, 773]
var_2 <= -0.848 gini = 0.518 samples = 192 value = [124, 0, 1, 0, 0, 0, 25, 42]
Ay_25 <= -1.676 gini = 0.009 samples = 227 value = [0, 226, 0, 0, 0, 0, 1, 0]
min <= -0.056 gini = 0.48 samples = 351 value = [3, 0, 26, 0, 0, 240, 76, 6]
gini = 0.0 samples = 358 value = [0, 0, 358, 0, 0, 0, 0, 0]
gini = 0.494 samples = 9 value = [0, 0, 4, 0, 0, 0, 5, 0]
Ay_11 <= 0.579 gini = 0.504 samples = 104 value = [60, 0, 0, 0, 0, 0, 2, 42]
var_2 <= 13.104 gini = 0.161 samples = 799 value = [31, 0, 1, 10, 11, 0, 15, 731]
gini = 0.523 samples = 60 value = [1, 0, 1, 0, 0, 0, 25, 33]
BA_DWT_6 <= -2.074 gini = 0.127 samples = 132 value = [123, 0, 0, 0, 0, 0, 0, 9]
gini = 0.0 samples = 226 value = [0, 226, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 0, 0, 1, 0]
var_3 <= -5.405 gini = 0.276 samples = 284 value = [1, 0, 23, 0, 0, 240, 15, 5]
gini = 0.168 samples = 67 value = [2, 0, 3, 0, 0, 0, 61, 1]
gini = 0.203 samples = 62 value = [55, 0, 0, 0, 0, 0, 1, 6]
gini = 0.251 samples = 42 value = [5, 0, 0, 0, 0, 0, 1, 36]
Stretch_FFT2 <= 0.065 gini = 0.13 samples = 783 value = [17, 0, 1, 10, 10, 0, 15, 730]
gini = 0.227 samples = 16 value = [14, 0, 0, 0, 1, 0, 0, 1]
gini = 0.0 samples = 4 value = [0, 0, 0, 0, 0, 0, 0, 4]
Ay_29 <= 0.96 gini = 0.075 samples = 128 value = [123, 0, 0, 0, 0, 0, 0, 5]
min <= -0.184 gini = 0.204 samples = 267 value = [0, 0, 23, 0, 0, 237, 5, 2]
gini = 0.588 samples = 17 value = [1, 0, 0, 0, 0, 3, 10, 3]
Stretch_FFT1 <= 0.02 gini = 0.105 samples = 772 value = [17, 0, 1, 10, 0, 0, 14, 730]
gini = 0.165 samples = 11 value = [0, 0, 0, 0, 10, 0, 1, 0]
BA_DWT_5 <= -1.723 gini = 0.016 samples = 123 value = [122, 0, 0, 0, 0, 0, 0, 1]
gini = 0.32 samples = 5 value = [1, 0, 0, 0, 0, 0, 0, 4]
gini = 0.0 samples = 181 value = [0, 0, 0, 0, 0, 181, 0, 0]
gini = 0.501 samples = 86 value = [0, 0, 23, 0, 0, 56, 5, 2]
var_1 <= -2.361 gini = 0.054 samples = 731 value = [6, 0, 0, 7, 0, 0, 7, 711]
gini = 0.678 samples = 41 value = [11, 0, 1, 3, 0, 0, 7, 19]
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 0, 0, 0, 1]
gini = 0.0 samples = 122 value = [122, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0 samples = 4 value = [0, 0, 0, 0, 0, 0, 4, 0]
Ax_16 <= -2.063 gini = 0.043 samples = 727 value = [6, 0, 0, 7, 0, 0, 3, 711]
gini = 0.562 samples = 11 value = [1, 0, 0, 6, 0, 0, 0, 4]
Ay_1 <= -0.821 gini = 0.025 samples = 716 value = [5, 0, 0, 1, 0, 0, 3, 707]
gini = 0.0 samples = 3 value = [3, 0, 0, 0, 0, 0, 0, 0]
Stretch_FFT2 <= -0.021 gini = 0.017 samples = 713 value = [2, 0, 0, 1, 0, 0, 3, 707]
gini = 0.444 samples = 3 value = [1, 0, 0, 0, 0, 0, 2, 0]
Ay_31 <= 5.206 gini = 0.008 samples = 710 value = [1, 0, 0, 1, 0, 0, 1, 707]
Ax_17 <= -2.094 gini = 0.006 samples = 709 value = [0, 0, 0, 1, 0, 0, 1, 707]
gini = 0.0 samples = 1 value = [1, 0, 0, 0, 0, 0, 0, 0]
gini = 0.444 samples = 3 value = [0, 0, 0, 1, 0, 0, 0, 2]
var_2 <= -5.237 gini = 0.003 samples = 706 value = [0, 0, 0, 0, 0, 0, 1, 705]
gini = 0.444 samples = 3 value = [0, 0, 0, 0, 0, 0, 1, 2]
gini = 0.0 samples = 703 value = [0, 0, 0, 0, 0, 0, 0, 703]

According Jupyter Notebook, accuracy of Decision Tree is 90.0 %. But if we want dou ble check we can use weka.



```
Feature set: 33,118,119,121

Time taken to build model: 3.97 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       2006               83.5833 %
Kappa statistic                         0.7767
Mean absolute error                     0.0964
Root mean squared error                 0.1949
Relative absolute error                50.5273 %
Root relative squared error            63.1395 %
Total Number of Instances            2400

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,951    0,183    0,775      0,951    0,854      0,752    0,954     0,921     Walk
                 0,418    0,012    0,696      0,418    0,522      0,516    0,880     0,444     Transition
                 0,929    0,006    0,973      0,929    0,950      0,939    0,992     0,976     Sit
                 0,539    0,021    0,758      0,539    0,630      0,605    0,913     0,659     Jump
                 0,772    0,014    0,879      0,772    0,822      0,802    0,976     0,855     Stand
                 0,900    0,007    0,946      0,900    0,922      0,913    0,988     0,947     Lie Down
                 0,000    0,000    0,000      0,000    0,000      -0,001   0,756     0,009     Stairs down
                 0,500    0,000    0,857      0,500    0,632      0,653    0,685     0,570     Stairs up
Weighted Avg.    0,836    0,079    0,835      0,836    0,827      0,778    0,956     0,863

=== Confusion Matrix ===
```

Accuracy with weka is 83.5%. And this result is not that close to our result. So we can use now weka for Random Forest classifier.

=== Summary ===

Correctly Classified Instances	2294	95.5833 %
Kappa statistic	0.9417	
Mean absolute error	0.0294	
Root mean squared error	0.104	
Relative absolute error	15.4029 %	
Root relative squared error	33.6763 %	
Total Number of Instances	2400	

As we see accuracy of Random Forest is 95%. Which means for this data set Random Forest is good for the classifying.

Conclusion

Taking everything into consideration, the main objective of the project is to show how human activity recognition can be used in different purposes. Our purpose was according to stretch and angle values, predict the current condition of user. For prediction, Decision Tree and Random Forest classifiers were used. To organize the data and use for our purposes we used Jupyter notebook. Through features that weka provide us, we can get accuracy and other information about data while classifying with Decision tree and Random Forest.