HUMAN ACTIVITY RECOGNITION

-Ankit Ranjan

INTRODUCTION

Samsung phones embedded with an accelerometer and gyroscope were used to capture tri-axial angular velocity, at 50 Hz, of people doing various activities like:

- Walking
- Walking upstairs
- Walking downstairs
- Sitting
- Standing
- Laying

We were provided with data of 30 people wearing the smartphone on their waists and performing the activities mentioned above. Now, the data collected consists of 561 different features generated by the raw accelerometer and gyroscope signals. Prediction using all the available features and data is a whopping 98%. But the overhead of generating that amount of data in real-time uses a lot of memory and processing power and is not feasible.

So, the aim of our project is to get highest possible accuracy in classification using least amount of features available. First hurdle is to cross the 80% prediction accuracy mark and then add minimum number of features to get prediction accuracy more than 90%. We managed to achieve our aim at the end of the project. The methods used and results obtained will be discussed in the later sections.

DATA COLLECTION AND PREPARATION

The data was provided to us in a '.rda' format, so the collection part was not done from our side. First of all, we converted the whole data to '.csv' format using Python is case we needed it in that format further on.

Before getting into the analysis part the column(features) names were first converted to unique names to avoid confusion using the make.names() method of R and the class value(activity feature) was stored in a separate array. For every model tested, we partitioned the dataset into two parts – Training(70%) and Testing(30%), for training and testing purposes respectively. So, the training data consists of observations pertaining to 21 subjects involved in the experiment and the remaining 9 make up the test data set.

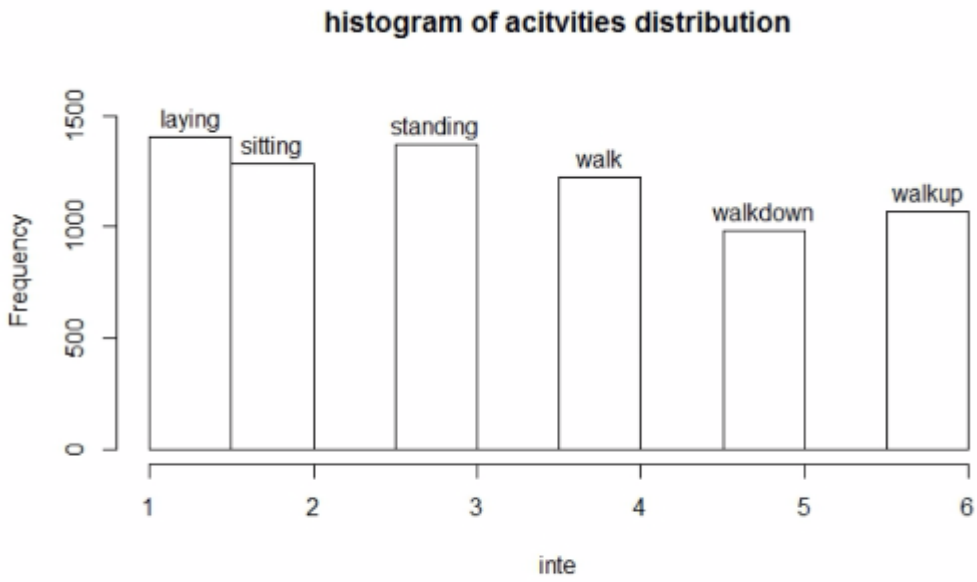

Figure 1. histogram of activity distribution

METHODS

The following models were used for our data analysis:

- Random Forest with Stepwise Multinomial Selections

  Random Forest is an ensemble learning method, meaning it combines multiple hypotheses to churn out one more general, and hopefully

better, hypothesis. A multitude of decision trees are formed with respect to all the data points and classes available at training time and then it finally outputs the class that is mode of the classes. Stepwise multinomial selection is a variable selection method that starts with no predictors (or features) and adds one predictor at every step. It is a greedy algorithm in the sense that it chooses the predictor that decreases the AIC (Akaike Information Criterion) the most at every step. AIC is used to determine the quality of statistical models with respect to a particular data set.

- LASSO

  For this method, first of all, the highly-correlated features were removed from the data set and then Lasso was applied to it. The LASSO is a regression method that involves penalizing the absolute size of the regression coefficients. By penalizing you end up in a situation where some of the parameter estimates may be exactly zero. The larger the penalty applied, the further estimates are shrunk towards zero. It is used to improve the prediction accuracy and interpretability of a statistical model.

ANALYSIS AND RESULTS:

- Random Forest with Stepwise Multinomial Selections

  The training data set formed earlier was used to create a random forest model for classification. This model was then used to get variable importance of the parameters. Of all the parameters top 50 were selected for further analysis. After that Stepwise Multimonial Selection was applied in order to further filter out the features and get a small subset of the features. As mentioned the stepwise selection method starts with zero predictors and adds predictors in a greedy fashion in order to get maximum reduction in AIC value. Our AIC lead to following results:

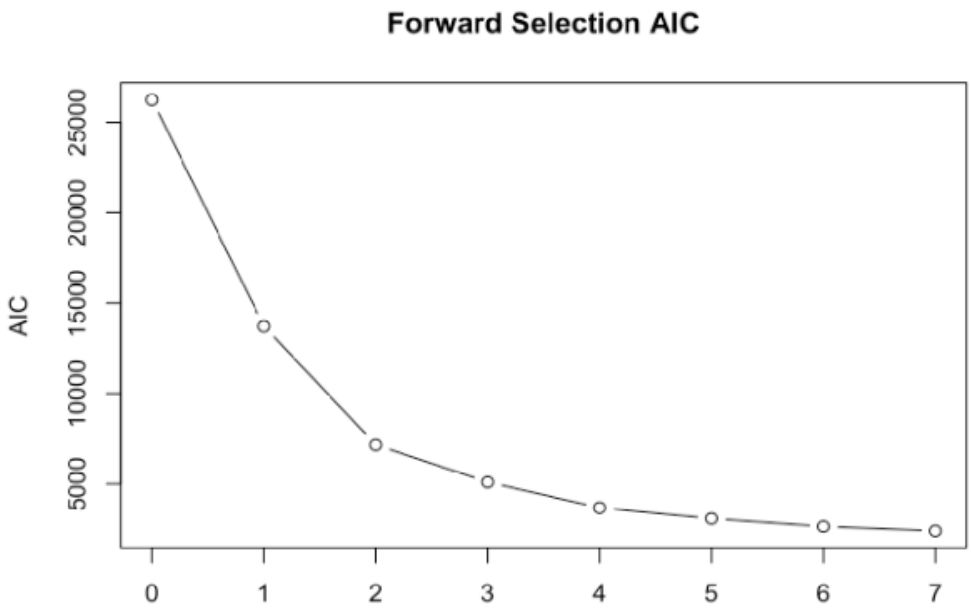

Figure 2. AIC Changes at Each Step

We restricted ourselves to 7 features and stopped the selection procedure after that to avoid increasing the complexity of the model. The features added at each step were as follows:

Step	Added Feature
0	Null Model
1	tBodyAcc.max...X
2	tGravityAcc.min...X
3	angle.Y.gravityMean.
4	tGravityAcc.arCoeff...X.1
5	tBodyAcc.correlation...X.Y
6	tGravityAcc.arCoeff...Y.2
7	fBodyGyro.meanFreq...X

Table 1. Features Added at Each Step

As we could see from Figure 2 the first three steps resulted in maximum drop in AIC value. There was still a worthwhile drop on the addition of the fourth feature but there wasn't a significant drop in AIC after the 4th step, or after addition of 4 predictors. So, we used the top 4 features shown in table 1 for our classification. A random forest model was built using these 4 features from the training data. Using this model 81.4% of

the test dataset was correctly classified. The confusion matrix is shown below:

Predicted	Ground Truth					
	Laying	sitting	Standing	Walk	Walkdown	walkup
Laying	537	0	0	0	0	0
Sitting	0	356	105	0	0	0
standing	0	135	426	0	0	0
Walk	0	0	0	440	53	98
Walkdown	0	0	0	14	302	35
Walkup	0	0	1	42	65	338

Table 2. Confusion Table for the StepAIC Result.

One notable thing about the matrix is that almost none (all except one) of the Standing/Sitting data was misclassified as any type of walking activity and none of the walking activities were classified as the former group at all.

- LASSO

Now for this model we first removed all the features that were highly correlated. We used Pearson Correlation method and any features with more than 0.95 correlation were removed. This was done to avoid overfitting our model. Using LASSO we first arrive at a value of lambda (severity of penalty), done with a 10-fold cross validation on traning set, that would be best suited for our model. The following figure shows the impact of lambda on goodness of fit,

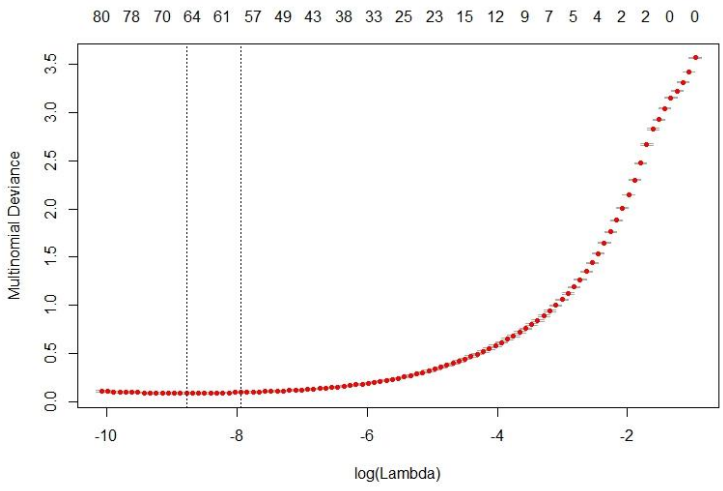

Figure 3. Choice of Lambda Influencing Model

Our focus is on Multinomial Deviance and the lower its value the better for our model. From the figure 3 we can see that the optimum solution for log-lambda is somewhere very close to log-lambda of (-8). This the value used in the final model that is built using LASSO. Now on final application of LASSO for model building we were left with a linear model with 6 non-zero parameters (5 predictors & 1 class value). The features and their respective parameters are as follows:

```
          (Intercept) tBodyAcc.correlation...Y.Z       tGravityAcc.mean...X        tGravityAcc.min...X
            8.5278321                -0.6972158                 -0.5291644                -15.3792106
tGravityAcc.energy...Y          tGravityAcc.iqr...X
            1.1176062                  1.0251588
```

Figure 4. Features and Their Parameters

Once done with model building we tested the performance of the model on our test data set. The following image displays the confusion matrix along with other measures like sensitivity, specificity, etc.

```
Prediction laying sitting standing walk walkdown walkup
  laying      526       0        0    0        0      0
  sitting       0     431       27    0        0      0
  standing     11      58      504    0        0      1
  walk          0       0        1  493        5     31
  walkdown      0       0        0    2      389      5
  walkup        0       2        0    1       26    434

Overall Statistics

               Accuracy : 0.9423
                 95% CI : (0.9333, 0.9505)
    No Information Rate : 0.1822
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9307
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: laying Class: sitting Class: standing Class: walk Class: walkdown Class: walkup
Sensitivity                 0.9795         0.8778          0.9474       0.9940          0.9262        0.9214
Specificity                 1.0000         0.9890          0.9710       0.9849          0.9972        0.9883
Pos Pred Value              1.0000         0.9410          0.8780       0.9302          0.9823        0.9374
Neg Pred Value              0.9955         0.9759          0.9882       0.9988          0.9878        0.9851
Prevalence                  0.1822         0.1666          0.1805       0.1683          0.1425        0.1598
Detection Rate              0.1785         0.1463          0.1710       0.1673          0.1320        0.1473
Detection Prevalence        0.1785         0.1554          0.1948       0.1798          0.1344        0.1571
Balanced Accuracy           0.9898         0.9334          0.9592       0.9894          0.9617        0.9549
```

Table 3. Summary of Goodness of Fit of Lasso Approach

The accuracy achieved using the features displayed in Figure 4, a total of 5 features, was a whopping 94.94% (~95%).

This time again Laying/Sitting/Standing data points were only classified once as one of the walking activities and vice versa was true too.

CONCLUSION

All in all the project was a success in the manner that we were able to achieve what we had set out to. Both the methods used gave varying results and with entirely different set of parameters. Maybe the covariance filter in LASSO helped us zero down to a better set of parameters.
Random forest method gave us 81.40% accuracy with 4 features, in the other hand LASSO gave us 94.94% (~95%) accuracy with 5, almost entirely different, features.

REFERENCES

1. "An Introduction to Feature Selection." Machine Learning Mastery. October 30, 2016.http://machinelearningmastery.com/an-introduction-to-feature-selection/.
2. Kuhn, Max. "The caret Package." http://topepo.github.io/caret/pre-processing.html#identifying-correlated-predictors.
3. CRAN - Package glmnet. (n.d.). RetrieveD from https://cran.r-project.org/web/packages/glmnet/index.html