Introduction:

This study is to build a function that predicts what activity a subject is performing based on the quantitative measurements from a smartphone. For this analysis we used the dataset complied on a Samsung Galaxy S II phone [1] and downloaded from the UCI dataset website [2]. We describe the study, our predictive model and **include relevant code snippets in gray background**.

Methods:

Data Collection

The data consists of measurements carried out on **21 unique subjects**. Different variables were measured using the gyroscope and the accelerometer of the smartphone and further data derviations (both in time and frequency domain) were carried out on these variables. In all 561 total variables were measure and are listed in the data set. The subjects noted the activity they were performing when the measurements were carried out. These activities were one of ("standing","sitting","laying","walk","walkdown","walkup"). The subject who was performing the activity was noted. In all 7352 observations are recorded. Thus the dataset has **7352 rows and 563 columns**.
In order to build a predictive model we divide the dataset into a train set and a test set. The test set includes all the data from subjects 27, 28, 29, and 30. The training set includes all the data from the remaining subjects (including subjects 1, 3, 5, and 6 as required by the assignment.) We were careful to make sure that the **train/test sets do not overlap**.

```
testset_sd <- sd[sd$subject%in%c(27,28,29),]
trainset_sd <- sd[!sd$subject%in%c(27,28,29),]
```

The README.txt [2] included with dataset gives an explanation of the variables measured and the dataset creation but a good explanation can also be found here [3]

Exploratory Analysis

Some of the measurement columns have duplicated names (yet contain unique information) which result in errors in downstream R commands. Since we did not want to throw out any unique information we decided to uniquify the names of the columns. We did this before we divided them into train/test set. We also changed the column names to a more readable format

```
names(sd) <- gsub("\\)","",names(sd))
names(sd) <- gsub("\\(","",names(sd))
names(sd) <- make.names(names(sd),unique =TRUE)
```

Finally we removed the subject column from the train and test set since it had no predictive value after the train/test were created.

```
testset_sd <- subset(testset_sd,select=-subject)
trainset_sd <- subset(trainset_sd,select=-subject)
```

There weren't any missing data/observations. As noted above in Data Collection, we created the train/test sets. We **did not create the validation set,** instead using all the remaining data in the train set. This is because the model that we finalize on is capable of doing cross-validation while it is working on the train set. After this data management **we have a train set with 6250 observations on 561 variables and a test set on 1102 observations and 561 variables**.

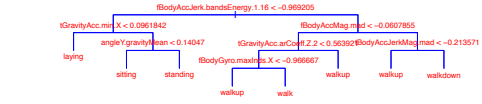
Statistical Modeling

The problem we are working on is a classification/prediction problem and not a linear regression problem. This suggests that the **relevant models to be used for prediction are trees and randomForests** (in classification mode). This also suggests that lm or glm models are likely to be less useful. We will thus concentrate our efforts trees and randomForests. We will briefly describe, without going into details, another more advanced method(Support Vector Machine).

Tree

The first classification model that we choose to evaluate is a normal tree model using all the variables for classification. It gives us a tree with 8 variables to split on. We then run cross validation on the tree and plot it to find the most relevant top variables. From the cross validation (**Figure 1**) we find that all 8 variables contribute to a reduction of deviation and hence we do not prune the tree. The tree that we get is shown below in **Figure1**

```
tree1 <- tree(as.factor(activity) ~ ., trainset_sd)
cv_tree <- cv.tree(tree1)
```



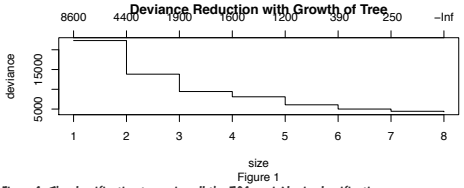


Figure 1

Figure 1: The classification tree using all the 561 variables in classification and the deviance reduction as function of the size of the tree.

The misclassification error rate of the tree on the training set is 10.21% as shown below:

```
Number of terminal nodes:  8
Residual mean deviance:  0.6382 = 3984 / 6242
Misclassification error rate: 0.1021 = 638 / 6250
```

randomForest[4]

Next we try a random forest on all the variables.

```
system.time(forest_sd <- randomForest(as.factor(activity) ~ ., data=trainset_sd, prox=TRUE, ntree=150, importance=T))

      OOB estimate of  error rate: 1.89%
Confusion matrix:
         laying sitting standing walk walkdown walkup  class.error
laying     1183       0        0    0        0      1 0.0008445946
sitting       0    1052       31    0        0      1 0.0295202952
standing      0      49     1101    0        0      0 0.0426086957
walk          0       0        0 1048        7      7 0.0131826742
walkdown      0       0        0    5      835      8 0.0153301887
walkup        0       0        0    3        6    913 0.0097613883
```

The misclassification error, confusion matrix and OOB error on the train set is also as given above. We use system.time to measure (**0.884 units of time**) the time it took to create the randomForest object. The importance variables are plotted in **Figure 2**. We will try to create a random Forest with just a limited number of the most important variables. We use the amount of time to create a randomForest object and the misclassification error to gauge the efficiency of reducing the # of variables used in creating an object. Once we have a new efficient randomForest with a limited numbers of variables and reasonable misclassification error we will use it to test on the test set.

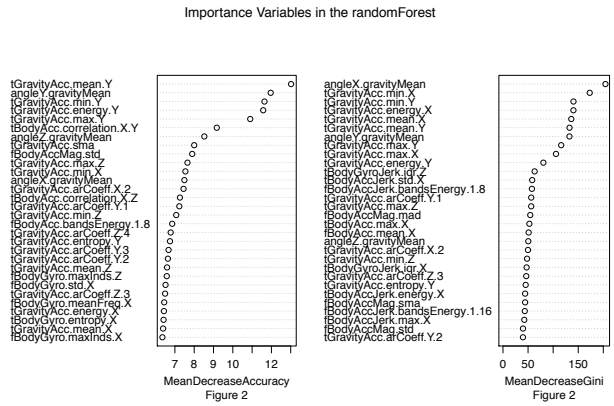


Figure 2: Importance Variables in the randomForest on 561 Variables.

From the above importance variables we select the most important variables (for all X,Y,Z coordinates) and run a randomForest a second time and create a confusion matrix. Again we use system.time to measure the system runtime (**0.469 units of time**). Thus we were able to reduce the # of variables from 561 down to ~ 30 variables and improving the run time by close to 100% (halving the run time). The OOB error increased from 1.89% to 1.92% which is negligible penalty to pay for 2x increase in efficiency. We thus finalize on a randomForest model using the variables as described below.

```
system.time(forest_sd_2 <- randomForest(as.factor(activity) ~
tGravityAcc.min.X   + tGravityAcc.mean.X    +  tGravityAcc.max.X   + tGravityAcc.energy.X  +
tGravityAcc.min.Y   + tGravityAcc.mean.Y    +  tGravityAcc.max.Y   + tGravityAcc.energy.Y  +
tGravityAcc.min.Z   + tGravityAcc.mean.Z    +  tGravityAcc.max.Z   + tGravityAcc.energy.Z  +
angleX.gravityMean  + angleY.gravityMean    +  angleZ.gravityMean  +
fBodyAccJerk.energy.X  + fBodyAccJerk.energy.Y  + fBodyAccJerk.energy.Z  +
tBodyAcc.correlation.X.Y + tBodyAcc.correlation.X.Z + tBodyAcc.correlation.Y.Z +
tGravityAcc.arCoeff.X.1 + tGravityAcc.arCoeff.X.2 + tGravityAcc.arCoeff.X.3 + tGravityAcc.arCoeff.X.4 +
tGravityAcc.arCoeff.Y.1 + tGravityAcc.arCoeff.Y.2 + tGravityAcc.arCoeff.Y.3 + tGravityAcc.arCoeff.Y.4 +
tGravityAcc.arCoeff.Z.1 + tGravityAcc.arCoeff.Z.2 + tGravityAcc.arCoeff.Z.3 + tGravityAcc.arCoeff.Z.4 +
fBodyAcc.bandsEnergy.1.8, data=trainset_sd, prox=TRUE, ntree=150, importance=T))

       OOB estimate of  error rate: 1.92%
Confusion matrix:
         laying sitting standing walk walkdown walkup  class.error
laying     1183       1        0    0        0      0 0.0008445946
sitting       0    1071       13    0        0      0 0.0119926199
standing      0      44     1106    0        0      0 0.0382608696
walk          0       0        0 1032       19     11 0.0282485876
walkdown      0       0        0    9      828     11 0.0235849057
walkup        0       0        0    8        4    910 0.0130151844
```

Reproducibility

For the cross validation results and randomForest to be reproducible we need to set the seed for the random number generator. This is essential because these are stochastic (pseudo-random) processes and hence indistinguishable from truly random processes. However they are repeatable if the seed is set before running them. In all our experiments we have set the seed to:

```
set.seed(123)
```

Further it is also important the train and test set be created as we described above to have completely reproducible results. If the train set is has different # of subjects or observations the importance variables may change slightly resulting in a different model.

Results:

Finally after finalizing on the randomForest model we are ready to run the test set on this model and note the misclassification error.

```
> table1 <- table(predict(forest_sd_2,newdata=testset_sd,type="class"), testset_sd$activity)
> table1

           laying sitting standing walk walkdown walkup
  laying      223       0        0    0        0      0
  sitting       0     164       31    0        0      0
  standing      0      38      193    0        0      0
  walk          0       0        0  160        1      4
  walkdown      0       0        0    4      137      2
  walkup        0       0        0    0        0    145
> sum(table1[row(table1) != col(table1)])/sum(table1)
[1] 0.07259528
```

We do a similar analysis for the randomForest generated from all the variables and the tree generated from all the variables. The results are in the table below:

	Tree with all Variables	**Random Forest with all Variables**	**Random Forest with ~ 30 important Variable**s
Misclassification Error	11.7%	4.62%	7.25%
Run Time	?	0.884 units of time	0.469 units of time

Confounders: Confounders are variables that are correlated to both the outcome and the covariates. They are hard to detect but may affect analysis. In our dataset it is very likely that the coordinates variable are confounders (i.e if a variable is changing in "X" direction while doing some activity it is very likely that it is changing in "Y" and "Z" direction as well). Confounders are especially important if we are looking for a causal relationship. However since we are working on a classification/prediction problem, we did not spend time looking and correcting for confounders. However we were particular to include "X", "Y", "Z" measurements for every importance variable in our randomForest to account for confounders.

Conclusions

We carried out a study to create a model to predict an activity that a person is engaged in from the measurements of 561 variables from the gyroscope and accelerometer of a smartphone. We finalized on a randomForest model with ~ 30 most important variables (out of a total of 561 possible). The prediction error on the test set was 7.25% but was twice as efficient (comparing run times) than the default randomForest.

If the 7.25% prediction rate is not acceptable, we can explore Support Vector Machine(SVM) or Neural Network(NN) based models that will train on all the variables. We need to make sure that the models are not over-trained/over-fitted by performing k-fold cross-validation on the train set.

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine. International Workshop of Ambient Assisted Living (IWAAL 2012). Vitoria-Gasteiz, Spain. Dec 2012

[2] UCI, Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones). Accessed Mar/2/2013

[3] David T Wilcox, Background on accelerometers, gyros and FFTs**,** https://class.coursera.org/dataanalysis-001/forum/thread?thread_id=2771. Accessed Mar/6/2013

[4] Leo Breiman et al, Random Forests http://www.stat.berkeley.edu/~breiman/RandomForests/. Accessed Mar/8/2013.