**Assignment 2 – Building a Predictive Model for Samsung Activity Data**

**Introduction:**

Cell phones are ubiquitous these days, and getting smarter and smarter each day. Given the ongoing competition in the market, companies like Samsung are continuously trying to capture our attention by coming up with sophisticated products and improved features.

Activity Recognition (AR) in cell phones aims to identify the actions carried out by a person given a set of observations of itself and the surrounding environment. [1] Recognition can be accomplished, for example, by exploiting the information retrieved from inertial sensors such as accelerometers [2] Analyzing this data can lead to useful insights about the way people function, and can provide them with timely information to make important decisions.

Here, I performed a predictive analysis to predict the activity of various subjects based on the accelerometer measurements from their cell phones while they were performing physical activities (standing, sitting, laying, walking, walking down and walking up). I built 2 predictive models using Support Vector Machines (SVM) and Random Forest methods, and trained those using training data. I then predicted the activity of the subjects using the test data. Finally, I compared them to identify the better predictive model. My results suggest that the accuracy rate of prediction is slightly better with the Random Forest model than the SVM model.

**Methods:**

*Data Collection*
Samsung phone accelerometer data containing 7532 samples for 21 unique subjects were downloaded from Coursera's Data Analysis Assignment 2 webpage [7] on March 3, 2013 using the R programming language [3].

*Exploratory Analysis*
Exploratory analysis was performed by examining tables and plots of the observed data. I identified transformations to perform on the raw data, on the basis of plots and knowledge of the scale of measured variables. Exploratory analysis was used to (1) identify missing values, (2) verify the quality of the data, and (3) determine the terms used in the predictive model relating activity and the accelerometer measurements.

*Statistical/Predictive Modeling and Reasons for Model Selection*
To predict the activity performed by a subject, I created 2 predictive models using SVM and Random Forest methods.

I chose SVM because its simplicity combined with state of the art performance on many learning problems (classification, regression, and novelty detection) has contributed to the popularity of the SVM. [5]

I chose Random Forest because
1) It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier. [4]
2) It can handle thousands of input variables without variable deletion. [4]

The 2 models were first trained using the training data suggested in the assignment page. [7] Due to memory limitations (8GB of RAM) and repeated computer crashes, the training set was limited to subjects 1, 3, 5, and 6 for both models. Subjects 25, 26, 27, 28, 29, and 30 were considered for the test set for both models. Model tuning was then performed for both, and a suitable prediction model function for each method was selected based on the lowest Out-Of-Band Error Rate Estimate (OOB) for Random Forest, and the lowest misclassification error rate for SVM. Finally, prediction was done for the test set and the results noted.

**Analysis and Results:**
The data was first scanned for possible NA values. There were none. The following analysis was done using the following models. All variables are considered, including confounders for both models since both models select variables automatically.

**SVM Model**
The SVM model maps the input data into a high-dimensional feature space defined by a kernel function. [5] The tuning of this model was done by manipulating 2 variables – gamma and cost, used by the kernel function.

The SVM training function was first built using the R library e1071 with a starting value of gamma= 0.1 and cost = 10, as follows:

```
svmTrain <- svm(as.factor(activity) ~ ., data=trainSet, cost = 10, gamma = 0.1)
summary(svmTrain)
```

where trainSet represents the training set with subjects 1, 3, 5, 6, and activity belongs to this set. svmTrain represents the SVM model.

The summary showed 1315 total support vectors or training patterns created for each activity. Variable selection happens automatically as svm() selects appropriate variables for training.

The SVM model was then tuned for the training data with tune.svm() for a range of values of gamma and cost. The cross validation parameter was included with cross=10 to perform cross validation.

The e1071 package tune.svm() function is a very useful function that suggests the best possible combination of gamma and cost, along with the misclassification error rates for each. I have included it below:

```
svmTuner <- tune.svm(as.factor(activity) ~ ., data = trainSet, gamma=10^(-6:-3), cost = 10^(1:2),
                cross = 10) # cross – indicates cross-validation
```

Here, gamma has 4 values ranging from 10^-6 to 10^-3, and cost has 2 distinct values of 10 and 100. svmTuner represents the tuned model function.

One drawback of using the tune.svm() function was that it took a very long time to execute. Initially, I had included many more subjects for the training set, but the computation took a lot of memory and resources, and crashed the computer. So I had to limit the training set to just 4 subjects as indicated in the assignment.

The summary of tuning was included as follows:

*summary(svmTuner)*

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
 0.001   10

- best performance: 0.009888966

- Detailed performance results:
  gamma cost       error  dispersion
1 1e-06   10 0.628920888 0.048720182
2 1e-05   10 0.108003701 0.019290133
3 1e-04   10 0.026642378 0.010973901
4 1e-03   10 0.009888966 0.006267519
5 1e-06  100 0.107246125 0.020478795
6 1e-05  100 0.026642378 0.010973901
7 1e-04  100 0.015209345 0.009490108
8 1e-03  100 0.009894749 0.006280753
```

As noted above, the best performance (in **bold**) was for a combination of gamma = 10^-3 and cost = 10. It resulted in the lowest misclassification error rate of 0.009888966.

The best values of gamma and cost were then collected as follows and supplied to the final prediction function svnTunedTrain() as follows:

```
bestGamma <- svmTuner$best.parameters[[1]] # gamma =10^-3
bestCost <- svmTuner$best.parameters[[2] # cost = 10

svmTunedTrain <- svm(as.factor(activity) ~ ., data = trainSet,
            cost = bestCost, gamma = bestGamma, cross = 10)
summary(svmTunedTrain)
```

where bestGamma indicates the best value of gamma, bestCose indicates the best value of cost. The summary indicated a total prediction accuracy of 99.23% on a 10-fold cross validation.

Next, the prediction function svnTunedTrain() was used for predicting the test set data as follows:

```
svmTunedPredict <- predict(svmTunedTrain, testSet)
```

where testSet indicates the test set for subjects 25,26,27,28,29,30.

A table named svmTable was created to compare the activity predicted by the predicted function svmTunedPredict with the actual activity observed, in the test set.
*svmTable <- table(observed= testSet$activity, svmPredict= svmTunedPredict)*
*svmTable*

```
          svmPredict
observed   laying sitting standing walk walkdown walkup
  laying      436       1        5    0        0      0
  sitting       9     304       94    0        0      0
  standing      0      10      421    0        0      0
  walk          0       0        0  286        2     74
  walkdown      0       0        0    1      207    100
  walkup        0       0        4    0        0    332
```

As seen in the table above, there were near perfect predictions for laying, walkdown, and walking. The classAgreement() function was used to get the accuracy of predictions.
*classAgreement(svmTable)*

```
$diag
[1] 0.8687664

$kappa
[1] 0.8418641

$rand
[1] 0.9231999

$crand
[1] 0.7348678
```

The diag variable indicated that there was 86.87% accuracy in the prediction.

**Random Forest Model**
Random Forest is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. [4] I used the tuning parameter *ntree* to specify the total number of trees.

The R library randomForest was used to construct a basic training function, considering all variables:
*rfTrain <- randomForest(as.factor(activity) ~ ., data=trainSet, prox=TRUE)*
where rfTrain is the training Random Forest function on the training set.

As observed, the total trees considered by the function was 500, and it used 23 variables for each classification tree. Variable selection for random forest was automatically determined. Cross-validation is not necessary to get a non-biased estimate, as it is estimated internally during the run. [6] The Out-Of-Band (OOB) error rate estimate was 0.68%.

The prediction function was tuned for different ntree values from 400 to 1000, and the following

function rfTrain with ntree = 700 was selected, as it had the lowest OOB error rate estimate of 0.46%. The confusion matrix showed an error rate of 0 for laying.

***rfTrain <- randomForest(as.factor(activity) ~ ., data=trainSet, prox=TRUE, ntree=700)***

Prediction was then done using rfTrain with the test set as follows, and a table rfTable was constructed to verify the predictions stored in rfPredict against the observed activity.

*rfPredict <- predict(rfTrain, testSet)*
*rfTable <- table(observed=testSet$activity, predict = rfPredict)*
*rfTable*

```
          rfPredict
observed   laying sitting standing walk walkdown walkup
  laying      442       0        0    0        0      0
  sitting       0     355       52    0        0      0
  standing      0      67      364    0        0      0
  walk          0       0        3  346        6      7
  walkdown      0       0        0    3      247     58
  walkup        0       0       41    0       25    270
```

As we can see, there was a perfect prediction for laying, and near-perfect one for walking. The classAgreement function was finally run to get the accuracy.

*classAgreement(rfTable)*

```
$diag
[1] 0.8853893

$kappa
[1] 0.8618666

$rand
[1] 0.9347213

$crand
[1] 0.7688228
```

The diag variable indicates that there was 88.53% accuracy in the prediction.

The final results of both the predictions was plotted in Figure 1. As we can see, there are 3 plots:

1) Left Panel - Basic plot of the test set activities per subject
2) Middle Panel - SVM prediction plot indicating the total predicted values of activity per subject
3) Right Panel - Random Forest prediction plot indicating the total predicted values of activity per subject

**Conclusions:**

Predictive analysis was done using SVM and Random Forest to identify a prediction model function that would accurately predict the activity of each subject for the test set given the accelerometer measurements. Based on the above analysis, the Random Forest prediction model was found to have a slightly better chance (88.53%) of prediction than the SVM model (86.87%).

My analysis suggests that Random Forest method is quite accurate, and can handle datasets with large number of variables. Although SVM model had a training accuracy of 99.23%, it was still unable to predict as accurately as I hoped for. It is possible that with a larger training set, the SVM prediction model may have outperformed the Random Forest model.

While this analysis is an interesting first step for similar Activity Recognition (AR) experiments, it is based on a limited sample of data available for the assignment with a limited number of activities. A larger collection of representative data may be more appropriate for creating a better prediction model. My analysis may be of interest to scientists and students seeking to construct predictive models using SVM and Random Forest.

**References:**


1. Davide Anguita1, Alessandro Ghio, Luca Oneto, Xavier Parra2, and Jorge L. Reyes-Ortiz: Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine. URL: http://www.icephd.org/sites/default/files/IWAAL2012.pdf, Accessed 3/8/2013.
2. Allen, F.R., Ambikairajah, E., Lovell, N.H., Celler, B.G.: Classication of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models. Physiological Measurement 27(10) (2006) 935. URL: http://iopscience.iop.org/0967-3334/27/10/001, Accessed 3/8/2013.
3. R Core Team (2012). ”R: A language and environment for statistical computing.” URL: http://www.R-project.org, Accessed 3/9/2013.
4. Wikipedia “Random Forest” Page. URL: http://en.wikipedia.org/wiki/Random_forest, Accessed 3/9/2013.
5. Karatzoglou A, Meyer D, Hornik K: Support Vector Machines in R. Journal of Statistical Software 2006. URL: http://www.jstatsoft.org/v15/i09/paper, Accessed 3/9/2013.
6. The out-of-bag (oob) error estimate. URL: http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#ooberr, Accessed 3/9/2013.
7. Samsung Accelerometer data. URL: https://class.coursera.org/dataanalysis-001/human_grading/view/courses/294/assessments/5/submissions**.** Accessed 2/9/2013.