Exercise Training with HAR

A Kissinger

11/20/2020

Synopsis

Human Activity Recognition (HAR) is a fledgling area in machine learning with tremendous possibilities. In this paper a model for detecting correct execution of an exercise is described. The training data was collected by Velloso *et al.* whose work is described in the paper *"Qualitative Activity Recognition of Weight Lifting Exercises"* listed in the bibliography at the end of this document. For the data collection, Velloso *et al* had six participants perform 10 repetitions of the Unilateral Dumbbell Biceps curl in five different ways: one according to the correct specification and the other four were performed in accordance with common mistakes. The participants had four sensors attached during these exercises; to their glove, armband, lumbar belt and dumbbell, each with 3-axes acceleration, gyroscope and magnetometer data. From these measurements Velloso *et al.* extracted eight features for each of the 3-axes, resulting in 96 derived feature sets for each participant(Velloso *et al.*, 2013).

After cleaning and removing all columns of data with NA content $> 95\%$, the remaining 60 features were used to train both an rpart and random forest model. The random forest model yielded the more accurate model with 99.4% accuracy and a .43% OOB error rate on the validation set.

Data Cleaning and Exploration

The data is read in and non-numerical character strings are recoded as 'NA' values; during the first modeling trial, it was found that many of the elements of the dataframe were "" (empty character strings) or "#DIV/0!" values.

```r
training <- read.csv("pml-training.csv",
    na.strings = c("", "#DIV/0!",
        "NA"))
testing <- read.csv("pml-testing.csv",
    na.strings = c("", "#DIV/0!",
        "NA"))
```

Exploration of the data through looking at the structure, *str(training)*, and running *colSum(is.na(training))* made it apparent that many of the columns were full of 'NA' values, including the values recoded in our last step. To take care of this, the columns consisting of $> 95\%$ NA values were removed from both the testing and training data sets. The first seven columns of each dataset were removed as well as they were unnecessary to train our model.

```r
# removing the rows having 95%
# or more of their values as
# NA's
exerdata <- training[, !((colSums(is.na(training)))/(nrow(training)) >
    0.95)]
exertest <- testing[, !((colSums(is.na(testing)))/(nrow(testing)) >
    0.95)]
```

```
# removing the first 7 columns
# as they will not be used in
# model
exerdata <- exerdata[, 8:60]
exertest <- exertest[, 8:60]
```

After removal of these columns, there were no NA values left in our testing or training datasets.

```
trainingNA <- sum(is.na(exerdata))
testingNA <- sum(is.na(exertest))
```

NA values in training data= 0.
NA values in testing data= 0.

The one factor variable in our dataset, the dependent variable classe, was coded as a factor for our models to work properly.

```
exerdata$classe <- as.factor(exerdata$classe)
```

Cross Validation

Accuracy based solely on the training data is optimistic and biased towards the data within the training set itself; this is known as resubstitution accuracy. Cross-validation splits the training data into training and validation sets before the model is trained to assess model accuracy on a separate set of data before it is used on the testing set; this is known as prediction accuracy. Here the training data is split into 75% training and 25% validation sets to perform cross-validation on our models.

```
set.seed(186)
inBuild <- createDataPartition(y = exerdata$classe,
    p = 0.75, list = FALSE)
exervalid <- exerdata[-inBuild,
    ]
exertrain <- exerdata[inBuild,
    ]
```

Tree Model Training and Validation

The first model built was a tree model using rpart with *tuneLength=7*. Here tunelength tries seven different configurations of the rpart algorithm to see which one works best, in the case of rpart, the tuning paramter is cp(complexity parameter). This is known as hyper-parameter tuning.

```
rpart <- train(classe ~ ., method = "rpart",
    data = exertrain, tuneLength = 7)
predrpart <- predict(rpart, newdata = exervalid)
cmatrix <- confusionMatrix(predrpart,
    exervalid$classe)$overall
cmatrix
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.490620e-01   5.581305e-01   6.355167e-01   6.624296e-01   2.844617e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00  2.596929e-188
```

Unfortunately the highest predicted accuracy for this model is 64.91%, and therefore it was decided to try the more robust but computationally intensive random forest method.

Random Forest Model Training and Validation

A random forest model was built in an attempt to obtain better prediction accuracy on the validation set.

Two optimizations were performed when training this model:
1). Number of trees(*ntree*): Values from 100-1000 in increments of 100 were tested to minimize the OOB (Out-of-Bag) error rate; OOB error rate minimization was found to occur at *ntree=500*.
2). Number of predictor variables used at each node (m): Values of m=sqrt(n), 2sqrt(n), sqrt(n)/2 (where n=number of predictors in the dataset) were tested for *m.try* to minimize the OOB error rate. The default *m.try=sqrt(n)* was found to be the best for minimizing OOB error rate.

```
set.seed(123)
rf <- randomForest(classe ~ .,
    data = exertrain, ntree = 500)
predrf <- predict(rf, newdata = exervalid)
cmatrixrf <- confusionMatrix(predrf,
    exervalid$classe)$overall
cmatrixrf
```

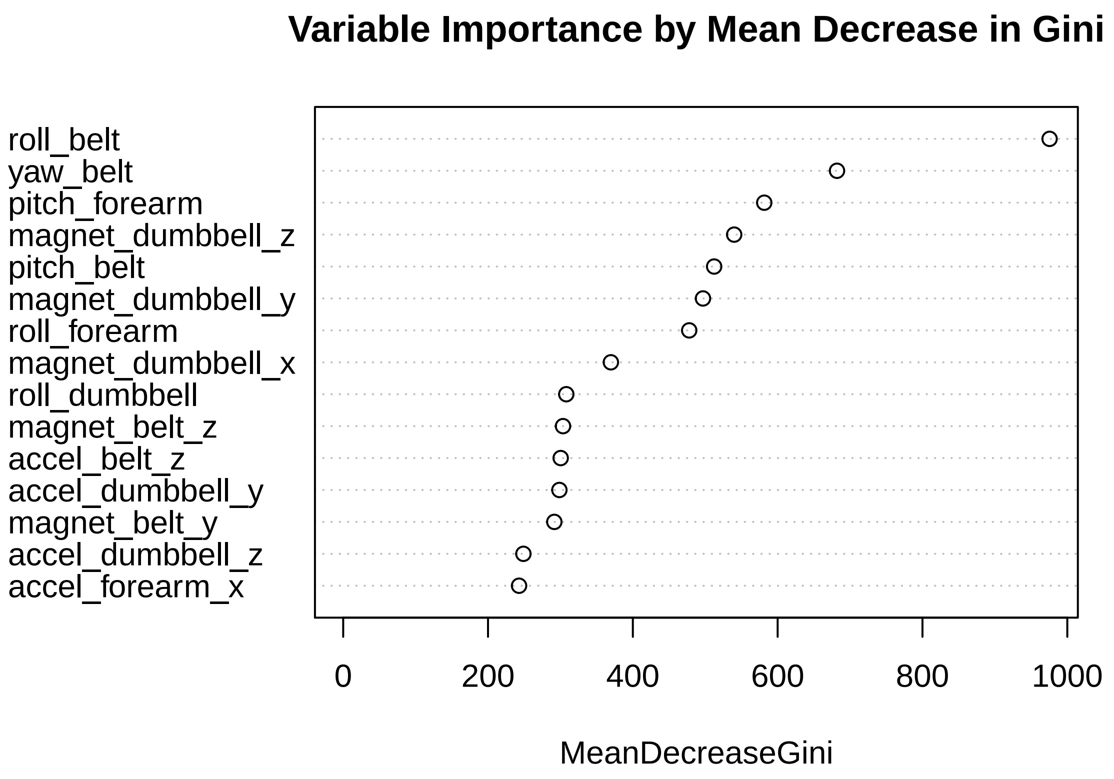
```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9938825      0.9922605      0.9912784      0.9958689      0.2844617
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
rf$err.rate[500, 1]  #OOB error rate for 500th tree
```

```
##         OOB
## 0.004348417
```

Using the random forest method produced an accuracy of 99.39% with an OOB error rate of .43% on our validation data. It was decided to move forward with the random forest as these rates are indicative of an excellent model. For a great explanation of OOB error rate see: https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710
Below is a plot depicting the importance of the predictor variables. The predictors are plotted against the mean decrease in Gini, where a Gini is a measure of how well a node partitions the data.

```
imprf <- importance(rf)
impPlotrf <- varImpPlot(rf, n.var = min(15,
    nrow(imprf)), main = "Variable Importance by Mean Decrease in Gini")
```



Model Predictions with Testing Set

After running both the rpart and random forest models on the validation data, the random forest model was the clear winner. The random forest model was then run on the testing set and the predicted values are shown below.

```r
predrffinal <- predict(rf, newdata = exertest)
predrffinal
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Bibliography

(1)Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.