Practical Machine Learning Course Project

Executive Summary

The following analysis attempts to use machine learning to correctly identify how well subjects perform specific activities. The data used in this study was collected as part of a study completed in 2012. More information about the dataset can be found on the publisher site.

The original dataset contains **19622** observations and **160** variables. The initial exploratory analysis indicated that the amount of variables can be reduced to **52**.

A random forest machine learning method was applied to the data which resulted in a promising model that seemed to indicate a high level of accuracy. This was confirmed when it was run against the test set and had a **99.48%** accuracy in predicting the 'classe' variable.

Data Preparation

Initial exploratory data analysis revealed some issues with the data that should be resolved before the analysis can take place. Due to the limitation of approximately 2000 words for this report the following section only provides an overview of any exploratory / data transformation steps taken.

Load appropriate libraries as well as the full data and validation sets

```
set.seed(1000);
library(caret)
library(randomForest)

setwd("~/R/predmachlearn")
ds = read.csv("pml-training.csv", as.is=TRUE)
validation = read.csv("pml-testing.csv")
```

```
#Assign the appropriate data types to variables
ds$classe <- as.factor(ds$classe)

#Convert all the remaining columns that are incorrectly classified as character to numeric
classes <- as.character(sapply(ds, class))
colClasses <- which(classes=="character")
ds[, colClasses] <- sapply(ds[, colClasses], as.numeric)
```

Remove all variables that do not have much variance. Some variables have more than 96% of the observations as NA. Find these columns and remove them from the dataset.

```
#Find all columns that have mostly NAs ie more than 96% of observations and then remove them
mostlyNasNames <- sapply(ds, function(x){ sum(!is.na(x)) > 19000})
mostlyNasNames <- names(mostlyNasNames[mostlyNasNames == TRUE])

ds.complete <- ds[, which(names(ds) %in% mostlyNasNames)]
ds.complete <- ds.complete[rowSums(is.na(ds.complete)) != ncol(ds.complete),]
```

Remove the first 5 columns X, user_name, raw_timestamp_part_1, raw_timestamp_part_2 and cvtd_timestamp as these variables are indexes or not related to the specific actions being performed ie. did not come from the measuring devices.

```
ds.complete <- ds.complete[,c(-1,-2,-3,-4,-5)]
```

Explicitly check for any variables that do not have much variance and remove them

```
nsv <- nearZeroVar(ds.complete, saveMetrics=TRUE)
nsv[nsv$nzv==TRUE,]
```

```
## [1] freqRatio     percentUnique zeroVar       nzv
## <0 rows> (or 0-length row.names)
```

Note that there are no variables with new zero values (True in the nzv column)

Data Analysis

The original dataset is first split into a training and testing set containing 60% and 40% of the observations respectively.

```
trainIndex <- createDataPartition(y=ds.complete$classe, p=0.6, list=FALSE)
train <- ds.complete[trainIndex, ]
test <- ds.complete[-trainIndex, ]
```

The following table provides an overview of the number of observations and the number of variables in each dataset.

____	Observations	Variables
Raw training set	19622	160
Raw validation set	20	160
Train (Incl. classe)	11776	52
Test (Incl. classe)	7846	52

Model Selection The model was trained using the a random forest (*method="rf"*) algorithm using the classe variable as the output and all the remaining variables as the predictors (*classe~.*). Out of bag (OOB) error re-sampling method with 10 re-sampling iterations (*method="oob", number=10*) was used to reduce the risk of overfitting the model instead of cross validation (Breiman, 1996).

```
#model <- train(classe~., data=train, method="rf", ntree=501, trControl=trainControl(method="oob", numbe

#save(model, file='RForest_Model')
```

```
#Load model from previous save
load('RForest_Model')
print(model$finalModel)
```

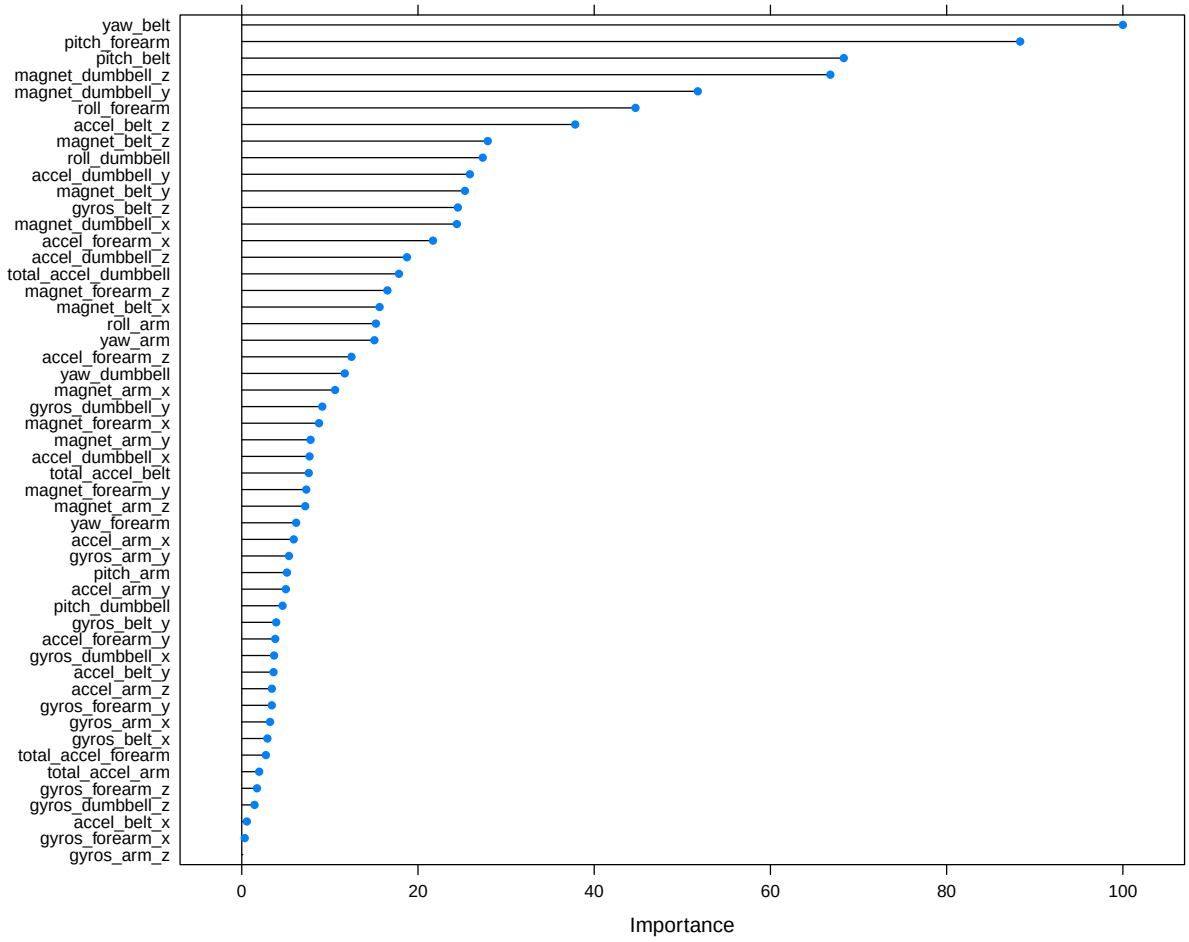
```
##
## Call:
##  randomForest(x = x, y = y, ntree = 501, mtry = param$mtry, proximity = TRUE,      do.trace = TRUE, a
##                Type of random forest: classification
##                      Number of trees: 501
```

```
## No. of variables tried at each split: 26
##
##         OOB estimate of  error rate: 0.91%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3338    7    1    1    1 0.002986858
## B   18 2252    8    0    1 0.011847301
## C    0   18 2029    7    0 0.012171373
## D    0    0   28 1900    2 0.015544041
## E    0    1    5    9 2150 0.006928406
```

The final model selected (above) indicate an misclassification or out-of-bag (OOB) error rate of only 0.91% which would indicate a very accurate model. The out-of-bag error rate is an accurate indication of how the model would perform on a test set of the similar size (Breiman, 1996b).

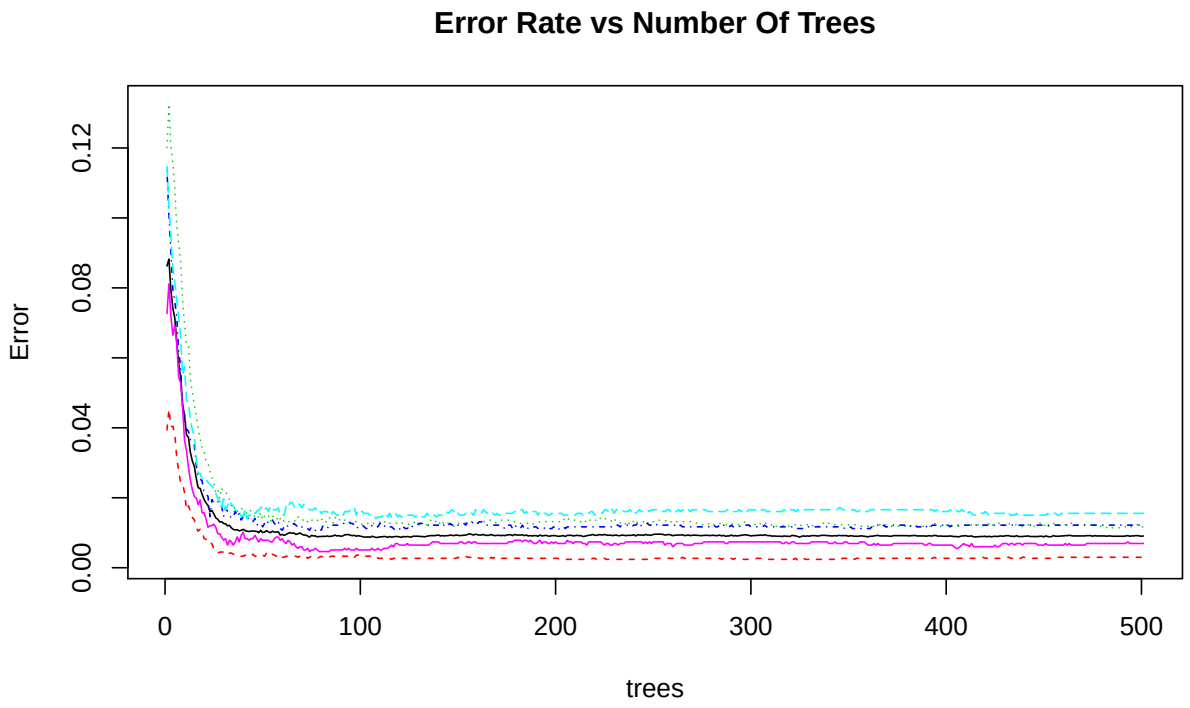
Due to the high accuracy of the random forest approach no other algorithms will be evaluated. That said looking at the variable importance in the next figure it is clear which variables are the most important. The model may be optimized for performance and interpretability by reducing the number of variables at the cost of accuracy.

```
plot(varImp(model))
```



Another optimization that can be experimented with is to reduce the number of trees used during each of the resampling iterations. Based on the following diagram indicates that the reduction in error stabilized at around 100 trees.

```
plot(model$finalModel, main = "Error Rate vs Number Of Trees")
```



Model Validation Validating the model using the test set confirms that the expected **out of sample error rate** is very close to the out of bag rate discussed in the previous section.

```
predictions <- predict(model, newdata=test)
cm <- confusionMatrix(predictions, test$classe)
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   15    0    0    0
##          B    1 1499    2    0    0
##          C    0    4 1364    9    0
##          D    0    0    2 1277    8
##          E    0    0    0    0 1434
##
## Overall Statistics
##
##                Accuracy : 0.9948
##                  95% CI : (0.9929, 0.9962)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9934
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9875   0.9971   0.9930   0.9945
## Specificity            0.9973   0.9995   0.9980   0.9985   1.0000
## Pos Pred Value         0.9933   0.9980   0.9906   0.9922   1.0000
## Neg Pred Value         0.9998   0.9970   0.9994   0.9986   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1911   0.1738   0.1628   0.1828
## Detection Prevalence   0.2863   0.1914   0.1755   0.1640   0.1828
## Balanced Accuracy      0.9984   0.9935   0.9975   0.9957   0.9972
```

The above output shows a confusion matrix providing an overview of the missclassified observations followed by the overall accuracy statistics. Note the high overall accuracy of 99.48%.

Conclusion Using a random forest machine learining algorithm created very accurate classification model with an accuracy of 99.48%.

Validation Set Prediction

The following provides an overview of the predicted classifications for the 20 observations in the validation dataset.

Remove all variables from the validation set that is not present in the training set and then run the predictions.

```
trainNames <- names(train)
validation.complete <- validation[, which(names(validation) %in% trainNames)]
validation.predictions <- predict(model, newdata=validation.complete)
```

Problem Id	Predicted Class
1	B
2	A
3	A
4	A
5	A
6	E
7	D
8	B
9	A
10	A
11	B
12	C
13	B
14	A
15	E

Problem Id	Predicted Class
16	E
17	A
18	B
19	B
20	B

```
#Write each prediction to a file that can later be submitted on the Coursera site.
#Iterate through the predictions and write to file
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
warnings()
pml_write_files(validation.predictions)
```

References

- Breiman, L. (1996). Out-of-bag estimation (pp. 1-13). Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996b. 33, 34.
- Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.