---

title: "Practical Machine Learning Course Project Report"

author: "Prasid"

date: "Sunday, October 25, 2015"

output: html_document

Practical Machine Learning Course Project Report

```{r}

library(ggplot2)

library(caret)

library(randomForest)

```

Load data

Load data.

Remove near zero covariates and those with more than 80% missing values since these variables will not provide much power for prediction.

Calculate correlations between each remaining feature to the response, classe. Use spearman rank based correlation because classe is a factor.

Plot the two features that have highest correlation with classe and color with classe to see if we can separate response based on these features.

```{r, echo=FALSE}

# load data
```

```
training <- read.csv("pml-training.csv", row.names = 1)

testing <- read.csv("pml-testing.csv", row.names = 1)

# remove near zero covariates

nsv <- nearZeroVar(training, saveMetrics = T)

training <- training[, !nsv$nzv]

# remove variables with more than 80% missing values

nav <- sapply(colnames(training), function(x) if(sum(is.na(training[, x])) >
0.8*nrow(training)){return(T)}else{return(F)})

training <- training[, !nav]

# calculate correlations

cor <- abs(sapply(colnames(training[, -ncol(training)]), function(x) cor(as.numeric(training[, x]),
as.numeric(training$classe), method = "spearman")))
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```{r, echo=FALSE}

# plot predictors

summary(cor)

plot(training[, names(which.max(cor))], training[, names(which.max(cor[-which.max(cor)]))], col =
training$classe, pch = 19, cex = 0.1, xlab = names(which.max(cor)), ylab = names(which.max(cor[-
which.max(cor)])))
```

The training set has r nrow(training) samples and r ncol(training) - 1 potential predictors after filtering.

There doesn't seem to be any strong predictors that correlates with classe well, so linear regression model is probably not suitable in this case. Boosting and random forests algorithms may generate more robust predictions for our data.

Boosting model

Fit model with boosting algorithm and 10-fold cross validation to predict classe with all other predictors.

Plot accuracy of this model on the scale [0.9, 1].

```{r, echo=FALSE}

set.seed(123)

boostFit <- train(classe ~ ., method = "gbm", data = training, verbose = F, trControl = trainControl(method = "cv", number = 10))

boostFit

plot(boostFit, ylim = c(0.9, 1))

```

The boosting algorithm generated a good model with accuracy = 0.997.

Random forests model

Fit model with random forests algorithm and 10-fold cross validation to predict classe with all other predictors.

Plot accuracy of the model on the same scale as boosting model.

```{r, echo=FALSE}

set.seed(123)
```

```
rfFit <- train(classe ~ ., method = "rf", data = training, importance = T, trControl = trainControl(method = "cv", number = 10))


rfFit

plot(rfFit, ylim = c(0.9, 1))

imp <- varImp(rfFit)$importance

imp$max <- apply(imp, 1, max)

imp <- imp[order(imp$max, decreasing = T), ]
```

The random forests algorithm generated a very accurate model with accuracy close to 1. Compared to boosting model, this model generally has better performance in terms of accuracy as we see from the plots.

Final model and prediction

Comparing model accuracy of the two models generated, random forests and boosting, random forests model has overall better accuracy. So, I'll use this model for prediction.

The final random forests model contains 500 trees with 40 variables tried at each split. The five most important predictors in this model are r rownames(imp)[1:5].

Estimated out of sample error rate for the random forests model is 0.04% as reported by the final model.

Predict the test set and output results for automatic grader.

```{r, echo=FALSE}
# final model

rfFit$finalModel

# prediction

(prediction <- as.character(predict(rfFit, testing)))
```

```
# write prediction files

pml_write_files = function(x){

  n = length(x)

  for(i in 1:n){

    filename = paste0("./prediction/problem_id_", i, ".txt")

    write.table(x[i], file = filename, quote = FALSE, row.names = FALSE, col.names = FALSE)

  }

}

pml_write_files(prediction)
```