# Machine Learning Prediction Assignment

*Kevin Goldsmith*

*9/22/2019*

Load relevant libraries, set seed

```
library(caret)
library(dplyr)
library(earth)
library(DataExplorer)
set.seed(1000)
```

Read in data

```
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

Simplify dataset during coding work, set to false for final run

```
simple = FALSE
if (simple == TRUE) {
    training <- sample_n(training, 100)
}
```

Explore data

```
plot_missing(training)
plot_histogram(training)
```

Clean data

```
# First 6 columns identify record, not predictive of classe so remove
training_clean <- training[, -c(1:7)]

# Identify and remove variables where records are primarily blank - no value
# in imputing variables that are mostly blank
bad <- intersect(which(apply(training_clean, 2, function(x) length(which(x ==
    "#DIV/0!")) > 0)), which(apply(training_clean, 2, function(x) length(unique(x)) <
    5)))
training_clean <- training_clean[-bad]

# All remaining variables are continuous, but many have been misidentified
# as factor variables. Reclass as numeric
classe <- training_clean$classe
training_clean <- lapply(training_clean[-79], function(x) as.numeric(as.character(x)))
training_clean <- as.data.frame(training_clean)
training_clean$classe = classe

# Some remaining columns are primarily NAs; remove these
completeCols <- which(colSums(is.na(training_clean)) == 0)
training_clean <- training_clean[completeCols]
```

Fit models, balancing parameter tuning to maximize resampling without taking prohibatively long

```r
# Fit simple tree model with 2 cross-validation iterations
fitControl <- trainControl(method = "cv", number = 2)

treeFit <- train(classe ~ ., data = training_clean, method = "rpart", trControl = fitControl)

# Fit bagging models with 10 iterations
bagFit <- train(classe ~ ., data = training_clean, method = "bagEarth", B = 10)


# Fit xgboost tree model with 50 iterations
parametersGrid <- expand.grid(eta = 0.3, colsample_bytree = 0.6, max_depth = 3,
    nrounds = 50, gamma = 0.5, min_child_weight = 2, subsample = 0.75)

trainX <- training_clean[-53]
trainY <- training_clean[53]
xgfit <- train(x = trainX, y = trainY$classe, method = "xgbTree", trControl = fitControl,
    tuneGrid = parametersGrid)

# Fit random forest model with k-fold cross-validation with k = 10 and 3
# iterations
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

rffit <- train(x = trainX, y = trainY$classe, method = "rf", trControl = fitControl,
    prox = TRUE)
```

Print accuracy of each model. Out of sample accuracy is expected to be slightly lower than accuracy of testing on training sets

```r
treepredicted <- predict(treeFit, training_clean)
treeMatrix <- confusionMatrix(treepredicted, training_clean$classe)

bagpredicted <- predict(bagFit, training_clean)
bagMatrix <- confusionMatrix(bagpredicted, training_clean$classe)

xgbpredicted <- predict(xgfit, training_clean)
xgbMatrix <- confusionMatrix(xgbpredicted, training_clean$classe)

rfpredicted <- predict(rffit, training_clean)
rfMatrix <- confusionMatrix(rfpredicted, training_clean$classe)

treeMatrix$overall[1]
```

```
##  Accuracy
## 0.4955662
```

```r
bagMatrix$overall[1]
```

```
##  Accuracy
## 0.7364693
```

```r
xgbMatrix$overall[1]
```

```
##  Accuracy
## 0.9726837
```

```r
rfMatrix$overall[1]
```

```
## Accuracy
##        1
```

Random forest model appears to be the most accurate, so select this one as final model and run on test data set

```
testPredict <- predict(rffit, testing)
print(testPredict)
```

```
##  [1] 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
```