PML Project

Synopsis:

After some initial exploratory analysis we are able to reduce the number of predictors down from 160 to 60 since there are many variables that consist of mostly missing data. The data is pretty varied and non normal so tree based models seem to be likely candidates. Decision Trees end up fitting poorly, and Random Forests at first seemed too good to be true. After removing some variables that may be overfitting the model we settle on a final Random Forest model with an Out of Sample Error expected to be around 5%. The 95% Confidence Interval for Accuracy was (94.25%, 95.64%) on this model.

Model selection Steps below

Load the data.

```
## Warning: package 'caret' was built under R version 3.0.3
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.0.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.0.3
```

```
## Warning: package 'rattle' was built under R version 3.0.3
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(123)
pth1 <- 'C:/Users/Signor/Desktop/Practical Machine Learning/'
trn.f <- 'pml-training.csv'
tst.f <- 'pml-testing.csv'
trn <- paste(pth1,trn.f, sep = '')
tst <- paste(pth1,tst.f, sep = '')

all.data <- read.csv (trn,header=T, sep=",");

cases <- read.csv (tst,header=T, sep=",");
```

```
str(all.data)
str(cases)
```

Take a quick look at the variables (output suppressed)

```r
inTrain = createDataPartition(all.data$classe, p = .6)[[1]]
training = all.data[ inTrain,]
remain = all.data[-inTrain,]

inTest = createDataPartition(remain$classe, p = .5)[[1]]
testing = remain[ inTest,]
validation = remain[-inTest,]
```

Partition the data into training and testing sets.

```r
na.percent <- rbind(sum(is.na(training[,1]))/nrow(training))
for(i in 2:160){
  na.percent[i] <- rbind(sum(is.na(training[,i]))/nrow(training))
}
table(na.percent)
```

Check the data to see if there are variables with a lot of missing information.

```
## na.percent
##                 0 0.978940217391304
##                93                67
```

```r
blank.percent <- rbind(sum(training[,1]== '')/nrow(training))
for(i in 1:160){
  blank.percent[i] <- rbind(sum(training[,i]== '')/nrow(training))
}
table(blank.percent)
```

```
## blank.percent
##                 0 0.978940217391304
##                60                33
```

```r
summary(training)
```

```r
tmp.remove <- as.data.frame(cbind(na.percent,blank.percent))
tmp.remove$drop <- ((tmp.remove$na.percent > .7) | (tmp.remove$blank.percent > .7))
table(tmp.remove$drop)
```

```
##
## FALSE  TRUE
##    60   100
```

```
drops <- as.data.frame(t(tmp.remove$drop))
var.names <- names(training)
names(drops) <- var.names
keep <- drops
for(i in 160:1){
  if (keep[,i] == TRUE) {
    keep[,i] <- NULL
  }
}

list <- names(keep)
use <- training[,list]
```

```
str(use)
summary(use)
```

100 variables have mostly missing data so they will be dropped, leaving 60 variables to begin with.

```
nsv <- nearZeroVar(use, saveMetrics = TRUE)
nsv
```

Checking to see if the remaining variables show little variance. (output suppressed)

```
n = ncol(use)
for(i in 60:1){
  if (is.numeric(use[,i])){
    hist(use[,i],xlab = paste('var',i,': ',list[i]),main = paste('type: ',class(use[,i])))
  }
  if (is.numeric(use[,i])==FALSE){
    plot(use[,i],xlab = paste('var',i,': ',list[i]),main = paste('type','(',nlevels(use[,i]),'): ',class
  }
}
n = ncol(use)
for(i in 59:1){
  if (is.numeric(use[,i])){
    plot(use[,i],use$classe,xlab = paste('var',i,': ',list[i]),main = paste('type: ',class(use[,i])))
  }
  if (is.numeric(use[,i])==FALSE){
    plot(use[,i],use$classe,xlab = paste('var',i,': ',list[i]),main = paste('type','(',nlevels(use[,i])
  }
}
```

Look at the plots of the variables to see what they look like (Normality/Scale etc.) to help decide what kinds of models to try. (output suppressed)

The data is pretty variable, and most of it does not appear to follow the normal distribution. A Decision Tree or Random Forest model will probably be the best algorithm.

```r
k.num <- 6
folds <- createFolds(y = use$classe, k=k.num, list = TRUE, returnTrain = FALSE)
fold1.train <- use[folds$Fold1,]
fold2.train <- use[folds$Fold2,]
fold3.train <- use[folds$Fold3,]
fold4.train <- use[folds$Fold4,]
fold5.train <- use[folds$Fold5,]
fold6.train <- use[folds$Fold6,]
```

Create folds in the training set to compare multiple models.

```r
modFit1 <- train(classe ~ ., method = "rpart", data = fold1.train)
```

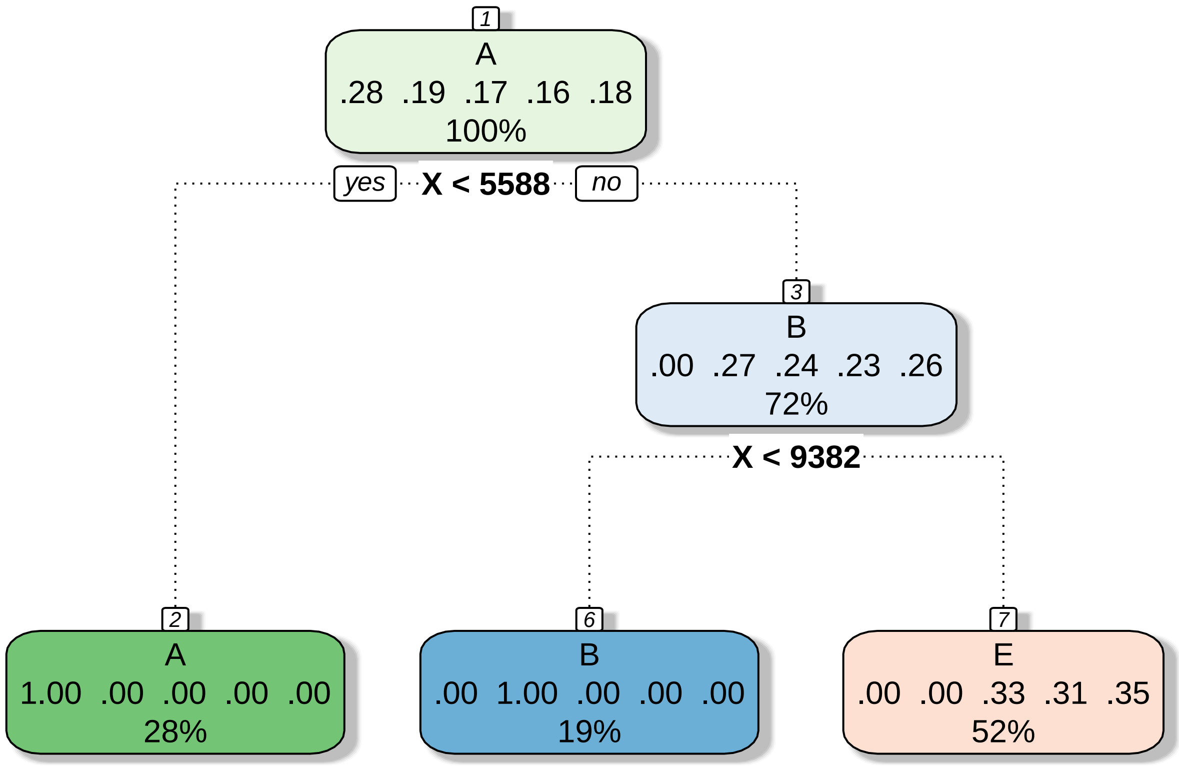
First model is a Decision Tree using all 60 variables.

```
## Loading required package: rpart
## Loading required namespace: e1071
```

```r
print(modFit1$finalModel)
```

```
## n= 1963
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 1963 1405 A (0.28 0.19 0.17 0.16 0.18)
##   2) X< 5588 558    0 A (1 0 0 0 0) *
##   3) X>=5588 1405 1025 B (0 0.27 0.24 0.23 0.26)
##     6) X< 9381.5 380    0 B (0 1 0 0 0) *
##     7) X>=9381.5 1025  664 E (0 0 0.33 0.31 0.35) *
```

```r
fancyRpartPlot(modFit1$finalModel)
```



Rattle 2015−May−26 08:20:17 Signor

```
p1 <- predict(modFit1,newdata = testing)
cM1 <- confusionMatrix(p1,testing$classe)
cM1$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    2    0    0    0
##          B    0  757    1    0    0
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E    0    0  683  643  721
```

```
cM1$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.6612287      0.5687988      0.6461749      0.6760419      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Pretty poor fit (66.12%)

```r
modFit2 <- train(classe ~ ., data = fold2.train, method = 'rf', prox=TRUE)
```

Second model is a Random Forest using all 60 variables:

```
## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.0.3

## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
varImp(modFit2)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 81)
##
##                                Overall
## X                                  100
## magnet_forearm_z                     0
## accel_arm_z                          0
## cvtd_timestamp02/12/2011 14:56       0
## accel_forearm_y                      0
## user_namepedro                       0
## gyros_belt_z                         0
## cvtd_timestamp28/11/2011 14:13       0
## magnet_arm_x                         0
## user_namecarlitos                    0
## gyros_forearm_x                      0
## accel_forearm_z                      0
## gyros_belt_x                         0
## yaw_arm                              0
## accel_arm_x                          0
## roll_forearm                         0
## gyros_arm_y                          0
## total_accel_dumbbell                 0
## gyros_dumbbell_z                     0
## cvtd_timestamp28/11/2011 14:15       0
```

```r
p2 <- predict(modFit2,newdata = testing)
cM2 <- confusionMatrix(p2,testing$classe)
cM2$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    0  759    1    0    0
##          C    0    0  677    0    0
##          D    0    0    6  643    0
##          E    0    0    0    0  721
```

```
cM2$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9982157      0.9977431      0.9963270      0.9992823      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Incredible accuracy (99.82%) may be the result of overfitting.

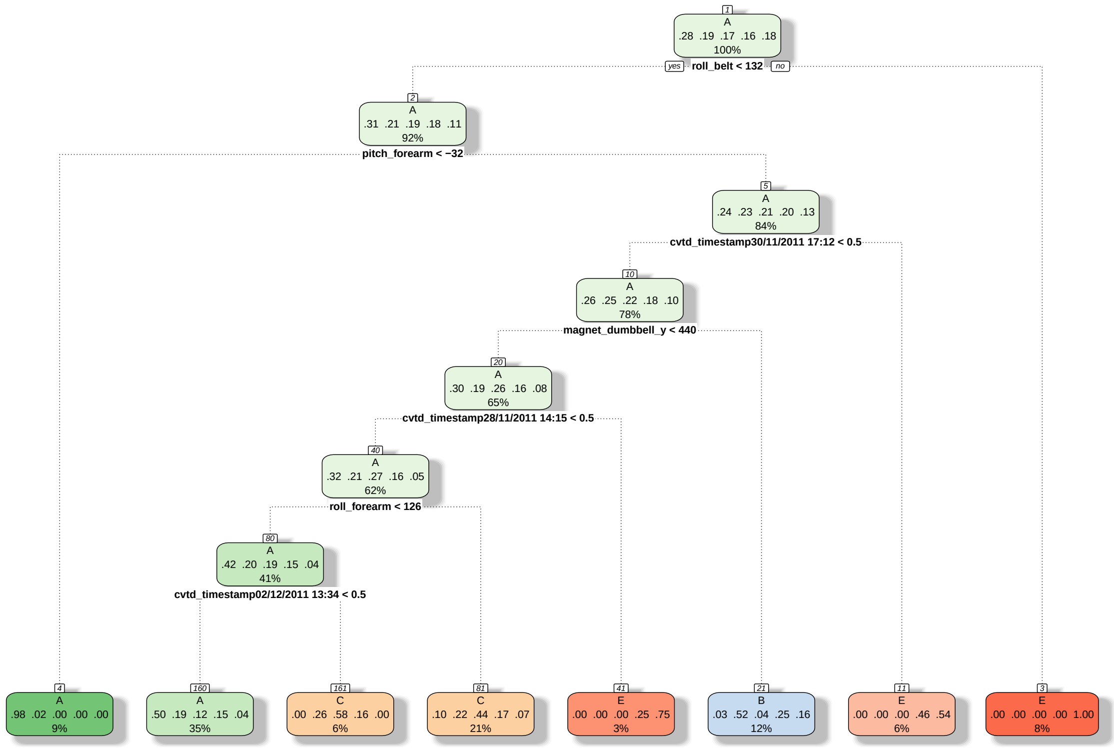
```
fold3.train$X <- NULL

modFit3 <- train(classe ~ ., method = "rpart", data = fold3.train)
print(modFit3$finalModel)
```

The 'X' varaible is the most important in both of these first two models, but is really just an arbitrary variable for the order in which the data was collected so we remove it from the data set and try another Decision Tree and Random Forest.

```
## n= 1963
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 1963 1405 A (0.28 0.19 0.17 0.16 0.18)
##     2) roll_belt< 131.5 1810 1252 A (0.31 0.21 0.19 0.18 0.11)
##       4) pitch_forearm< -32.1 169    3 A (0.98 0.018 0 0 0) *
##       5) pitch_forearm>=-32.1 1641 1249 A (0.24 0.23 0.21 0.2 0.13)
##        10) cvtd_timestamp30/11/2011 17:12< 0.5 1529 1137 A (0.26 0.25 0.22 0.18 0.097)
##          20) magnet_dumbbell_y< 439.5 1285  900 A (0.3 0.19 0.26 0.16 0.085)
##            40) cvtd_timestamp28/11/2011 14:15< 0.5 1217  832 A (0.32 0.21 0.27 0.16 0.048)
##              80) roll_forearm< 125.5 810  466 A (0.42 0.2 0.19 0.15 0.036)
##               160) cvtd_timestamp02/12/2011 13:34< 0.5 692  348 A (0.5 0.19 0.12 0.15 0.042) *
##               161) cvtd_timestamp02/12/2011 13:34>=0.5 118   50 C (0 0.26 0.58 0.16 0) *
##              81) roll_forearm>=125.5 407  228 C (0.1 0.22 0.44 0.17 0.071) *
##            41) cvtd_timestamp28/11/2011 14:15>=0.5 68   17 E (0 0 0 0.25 0.75) *
##          21) magnet_dumbbell_y>=439.5 244  117 B (0.029 0.52 0.041 0.25 0.16) *
##        11) cvtd_timestamp30/11/2011 17:12>=0.5 112   52 E (0 0 0 0.46 0.54) *
##     3) roll_belt>=131.5 153    0 E (0 0 0 0 1) *
```

```
fancyRpartPlot(modFit3$finalModel)
```



Rattle 2015−May−26 08:28:37 Signor

```r
p3 <- predict(modFit3,newdata = testing)
cM3 <- confusionMatrix(p3,testing$classe)
cM3$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1032  278  172  229   47
##          B   11  250   10   90   82
##          C   73  231  501  181   67
##          D    0    0    0    0    0
##          E    0    0    1  143  525
```

```r
cM3$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   5.883253e-01   4.671076e-01   5.727366e-01   6.037822e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00  1.262672e-278
```

```r
fold4.train$X <- NULL

modFit4 <- train(classe ~ ., data = fold4.train, method = 'rf', prox=TRUE)
varImp(modFit4)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 80)
##
##                               Overall
## raw_timestamp_part_1          100.000
## roll_belt                      67.861
## num_window                     53.857
## pitch_forearm                  42.783
## magnet_dumbbell_y              32.345
## magnet_dumbbell_z              29.713
## roll_forearm                   18.141
## pitch_belt                     16.792
## yaw_belt                       16.033
## cvtd_timestamp30/11/2011 17:12  14.035
## accel_forearm_x                13.052
## accel_belt_z                   12.154
## roll_dumbbell                  11.170
## magnet_dumbbell_x              10.604
## cvtd_timestamp02/12/2011 14:58  10.531
## cvtd_timestamp02/12/2011 13:33   9.656
## accel_dumbbell_y                9.605
## magnet_belt_y                   9.327
## cvtd_timestamp28/11/2011 14:15   9.217
## cvtd_timestamp05/12/2011 11:24   8.354
```

```
p4 <- predict(modFit4,newdata = testing)
cM4 <- confusionMatrix(p4,testing$classe)
cM4$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    1    0    0    0
##          B    2  753   16    0    0
##          C    0    5  667    5    0
##          D    0    0    1  637    5
##          E    0    0    0    1  716
```

```
cM4$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9908233      0.9883925      0.9873181      0.9935647      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

The Decision Tree model is not very good (58.83% accuracy) so it seems a Decision Tree may not be the way to go.

```
fold5.train$X <- NULL
fold5.train$user_name <- NULL
```

```
fold5.train$raw_timestamp_part_1 <- NULL
fold5.train$raw_timestamp_part_2 <- NULL
fold5.train$cvtd_timestamp <- NULL
fold5.train$new_window <- NULL
fold5.train$num_window <- NULL

modFit5 <- train(classe ~ ., data = fold5.train, method = 'rf', prox=TRUE)
varImp(modFit5)
```

The new Random Forest Model was still really good (99.08% accuracy) so that will be the type of model to use, but it may still be overfitting. The num_window and timestamp variables are some of the more important in the model but will not be relatable to data collected in the future. Let's try another Random Forest with just the measurement variables in the Data Set, removing the variables related to the user and time of the activity. This will probably most accurately reflect how well the model can do to predict data that happens in the future.

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          73.92
## yaw_belt               53.66
## magnet_dumbbell_z      47.80
## magnet_dumbbell_y      39.73
## roll_forearm           39.41
## pitch_belt             37.70
## roll_dumbbell          29.32
## accel_dumbbell_y       22.63
## magnet_belt_y          20.83
## magnet_dumbbell_x      20.26
## accel_forearm_x        19.52
## magnet_belt_z          19.19
## accel_dumbbell_z       15.98
## total_accel_dumbbell   14.99
## magnet_forearm_z       13.15
## gyros_dumbbell_y       12.51
## accel_belt_z           11.92
## yaw_dumbbell           11.51
## accel_forearm_z        11.40
```

```
p5 <- predict(modFit5,newdata = testing)
cM5 <- confusionMatrix(p5,testing$classe)
cM5$table
```

```
##             Reference
## Prediction    A    B    C    D    E
##          A 1089   37    4    7    1
##          B    8  702   27    2    5
##          C    6   15  651   24   13
##          D   12    4    2  606    9
##          E    1    1    0    4  693
```

```r
cM5$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.536069e-01   9.412805e-01   9.465516e-01   9.599762e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   2.665045e-09
```

Pretty good (95.36%) performance overall, so this will be the Final Model.

```r
pred.val <- predict(modFit5,newdata = validation)
confusionMatrix(pred.val,validation$classe)
```

Testing the Model on the validation set to estimate Out of Sample Error.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1094   31    1    1    1
##          B    6  704   41    2    5
##          C    4   19  638   39   12
##          D   10    2    4  597   10
##          E    2    3    0    4  693
##
## Overall Statistics
##
##                Accuracy : 0.9498
##                  95% CI : (0.9425, 0.9564)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9365
##  Mcnemar's Test P-Value : 1.228e-12
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9803   0.9275   0.9327   0.9285   0.9612
## Specificity            0.9879   0.9829   0.9772   0.9921   0.9972
## Pos Pred Value         0.9699   0.9288   0.8961   0.9583   0.9872
## Neg Pred Value         0.9921   0.9826   0.9857   0.9861   0.9913
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2789   0.1795   0.1626   0.1522   0.1767
## Detection Prevalence   0.2875   0.1932   0.1815   0.1588   0.1789
## Balanced Accuracy      0.9841   0.9552   0.9550   0.9603   0.9792
```

```r
cases.pred <- predict(modFit5,newdata = cases)
```

Predict values of 20 cases for submission.