ML_Project

James Martherus

10/9/2016

To begin, I loaded the data and packages we will need later:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

```
library(ggplot2)
library(reshape2)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

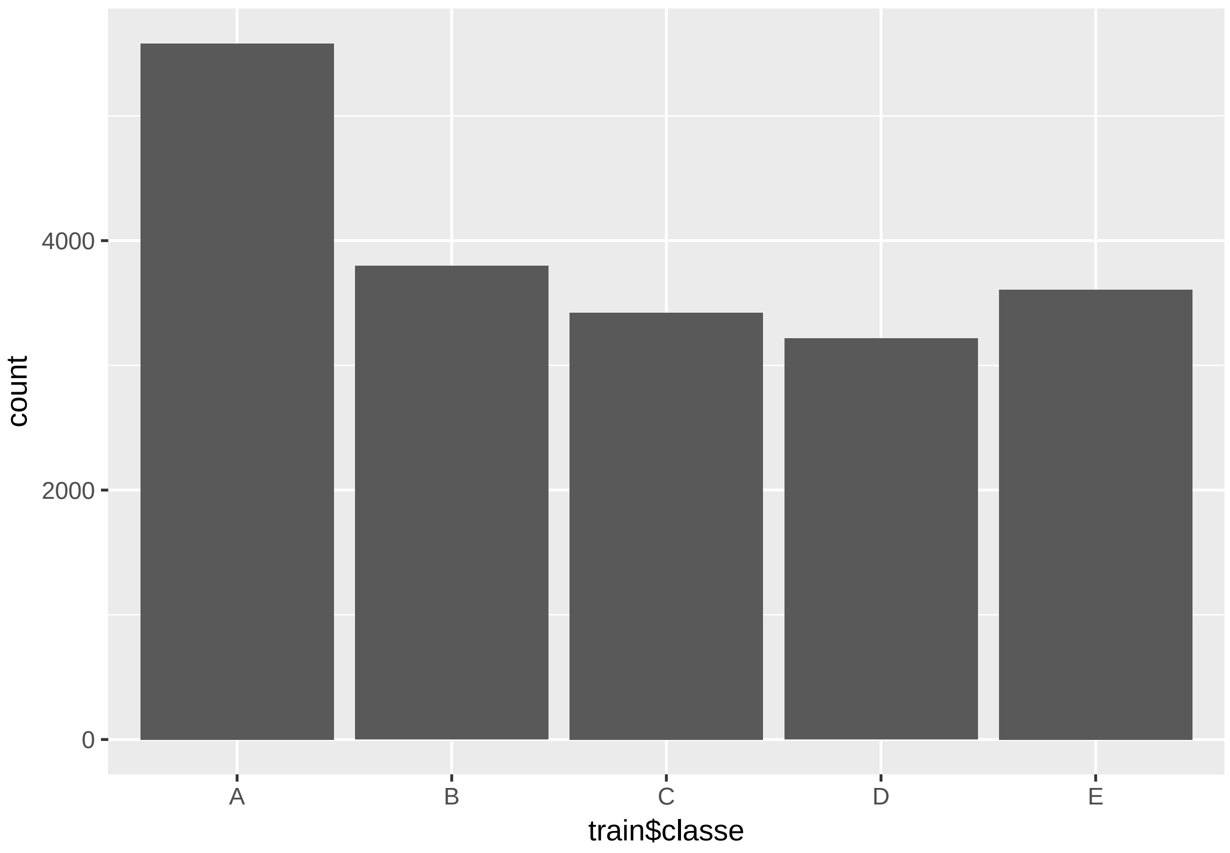
```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
train <- read.csv("/users/jamesmartherus/Documents/Coursera/Machine Learning/pml-training.csv")

test <- read.csv("/users/jamesmartherus/Documents/Coursera/Machine Learning/pml-testing.csv")
```

We then take a look at the data:

```
train$classe <- as.factor(train$classe)
qplot(train$classe)
```



It appears that the variable we are predicting has five categories, with a bias towards category A.

Next we should do some preprocessing. First we get rid of all variables with missing values. If the model performs poorly, we can always add them back in and impute the missing values, but this may be unneccessary. Second, we get rid of some of the first variables in the dataset which are obviously useless.

```r
#Get rid of vars with missing values
train <- train[, colSums(is.na(train)) == 0]
test <- test[, colSums(is.na(test)) == 0]

#Get rid of some obviously useless variables
training <- train[, -c(1:7)]
testing <- test[, -c(1:7)]

#Get rid of variables with no variance
myNZVvars <- names(training) %in% c("new_window", "kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "skewness_roll_dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_dumbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")
training <- training[!myNZVvars]

for (i in 1:length(testing) ) {
        for(j in 1:length(training)) {
        if( length( grep(names(training[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(training[i])
        }
    }
}
```

Then, we partition the training data into a true training set and a validation set. This allows us to gauge out of sample error without using our actual test data.

```r
inTrain <- createDataPartition(y=train$classe, p=.75, list=FALSE)
internal_train <- train[inTrain,]
internal_test <- train[-inTrain,]
```

Now we run a random forest model.

```r
set.seed(38576)
control <- trainControl(method = "cv", number = 5)
model1 <- train(classe ~ ., data = training, method = "rf",
                   trControl = control)
print(model1, digits = 4)
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15699, 15697, 15696, 15699, 15697
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9938    0.9922
##   27    0.9939    0.9923
##   52    0.9885    0.9854
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Next we use the model to predict our validation data. Since we have good accuracy, we can now test the model on our actual testing data.

```r
# predict outcomes using validation set
predict_rf <- predict(model1, internal_test)
# Show prediction result
(conf_rf <- confusionMatrix(internal_test$classe, predict_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    0  855    0    0
##          D    0    0    0  804    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9992, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
(accuracy_rf <- conf_rf$overall[1])
```

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

It appears that the random forest model predicts whether the exercise was performed correctly quite well.

```
(predict(model1, testing))
```

```
##  [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
```