Practical Machine Learning

Leo R

30 January 2019

Initial loadout of libraries

Data gathering, cleaning, and exploration

Gather the data from the provided URLs, and transform NAs to R-friendly NAs.

```
train_URL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
test_URL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
train_orig <- read.csv(url(train_URL),sep = ",", na.strings = c("",
"NA","#DIV/0!"))
test_orig <- read.csv(url(test_URL),sep = ",", na.strings = c("",
"NA","#DIV/0!"))
```

Initial look at the data - commented out due to its verbose nature.

```
#head(train_orig)
#head(test_orig)
```

Looking at the data we can see the first 8 columns are of little use to us (including sample number, dates, etc. which should not help prediction), and the dataset has many variables which are just filled with NAs. We can clean the dataframe to only include the relevant and non-NA variables below:

```
train_orig <- train_orig[, colSums(is.na(train_orig)) == 0]
test_orig <- test_orig[, colSums(is.na(test_orig)) == 0]

train_clean <- train_orig[,-c(1:8)]
test_clean <- test_orig[,-c(1:8)]
```

Model building

We will need to split the training set to train and cross validation data sets in order to perform cross validation. We will split the data 80-20.

```
set.seed(888)
cv_flag <- createDataPartition(train_clean$classe, p=0.80, list=F)
train_split <- train_clean [cv_flag, ]
cv_split <- train_clean[-cv_flag, ]
```

Now that we've split the data, we can train the model. We're using 10-fold cross validation to train a random forest model, and we've allowed parallel processing to speed up the process. [Note to anyone wanting to run the code: this takes a while to run]. After training the model, We'll generate a confusion matrix of itws performance against the "test split", and estimate its accuracy and out of sample error.

```
Mod1<-trainControl(method="cv", number=10, allowParallel=T, verbose=F)

rf_model<-train(classe~.,data=train_split, method="rf", trControl=Mod1,
verbose=F)

pred_rfm<-predict(rf_model, cv_split)

confusionMatrix(cv_split$classe, pred_rfm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    3  755    1    0    0
##          C    0    5  677    2    0
##          D    0    0   10  633    0
##          E    0    0    1    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9915, 0.9965)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9973   0.9934   0.9826   0.9969   1.0000
## Specificity            1.0000   0.9987   0.9978   0.9970   0.9997
## Pos Pred Value         1.0000   0.9947   0.9898   0.9844   0.9986
## Neg Pred Value         0.9989   0.9984   0.9963   0.9994   1.0000
## Prevalence             0.2852   0.1937   0.1756   0.1619   0.1835
## Detection Rate         0.2845   0.1925   0.1726   0.1614   0.1835
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9987   0.9961   0.9902   0.9969   0.9998
```

```
accuracy <- postResample(pred_rfm, cv_split$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9943920 0.9929057

out_of_sample_error <- 1 - as.numeric(confusionMatrix(cv_split$classe,
pred_rfm)$overall[1])
out_of_sample_error

## [1] 0.005607953
```

The acurracy for the model is 0.994, with a Kappa of 0.993. The out-of-sample-error for the model is approximately 0.006.

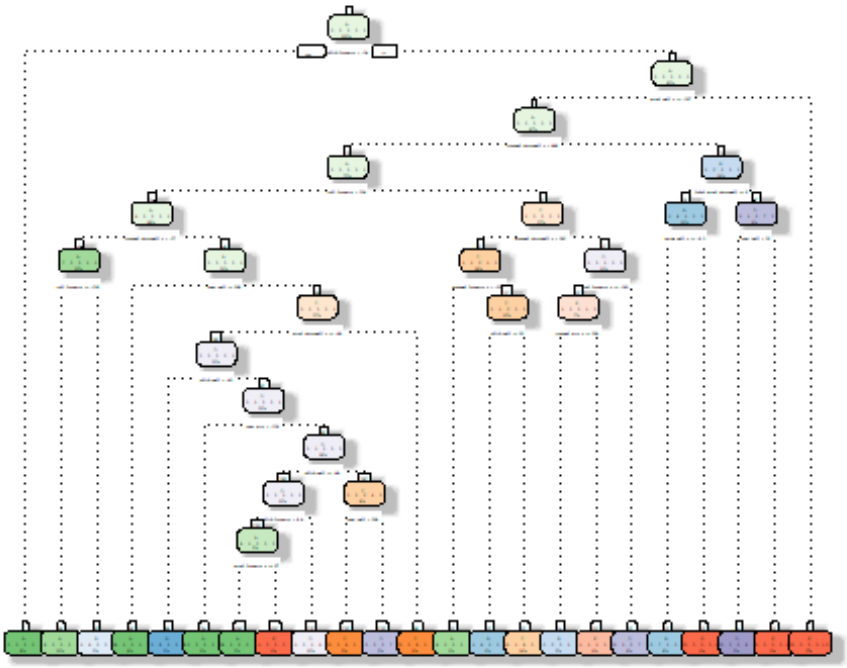
Now that we have trained a satisfactory model, we can use our trained model to predict the original 20 test cases.

```
predict_test_cases <- predict(rf_model, test_clean[, -
length(names(test_clean))])
predict_test_cases

##  [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
```

Appendix: Tree plot figure

Here we can illustrate the decision tree using rpart.plot

```
fancy_model <- rpart(classe ~., data=train_clean, method="class")
fancyRpartPlot(fancy_model, digits=1)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2019-Jan-30 16:43:11 Leo