Organizing Data

Clean and Import Data

Load the data by downloading the files and saving adding them. Training set is 19623X160 and test is 20X160

The scrubbed data is 19623*153 and 20*153. “classes as last”

```
#load the required packages
library(caret); library(rattle); library(rpart); library(rpart.plot)
library(randomForest); library(repmis)
#import data and load data from local file
training <- read.csv("C:\\projects\\practicleML\\pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("C:\\projects\\practicleML\\pml-testing.csv", na.strings = c("NA", ""))
#remove /null/empty values
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]

#remove first seven columns/predictors as the does not impact predictions
training_data <- training[, -c(1:7)]
test_data <- testing[, -c(1:7)]
```

Partition Data

In order to get out-of-sample errors, we split the cleaned training set `training_data` into a training set (`train`, 70%) for prediction and a validation set (`valid` 30%) to compute the out-of-sample errors.

```
set.seed(7826)
inTrain <- createDataPartition(training_data$classe, p = 0.7, list = FALSE)
train <- training_data[inTrain, ]
valid <- training_data[-inTrain, ]
```

RF and Classification Algos test

Using classification trees and random forest.

Classification trees

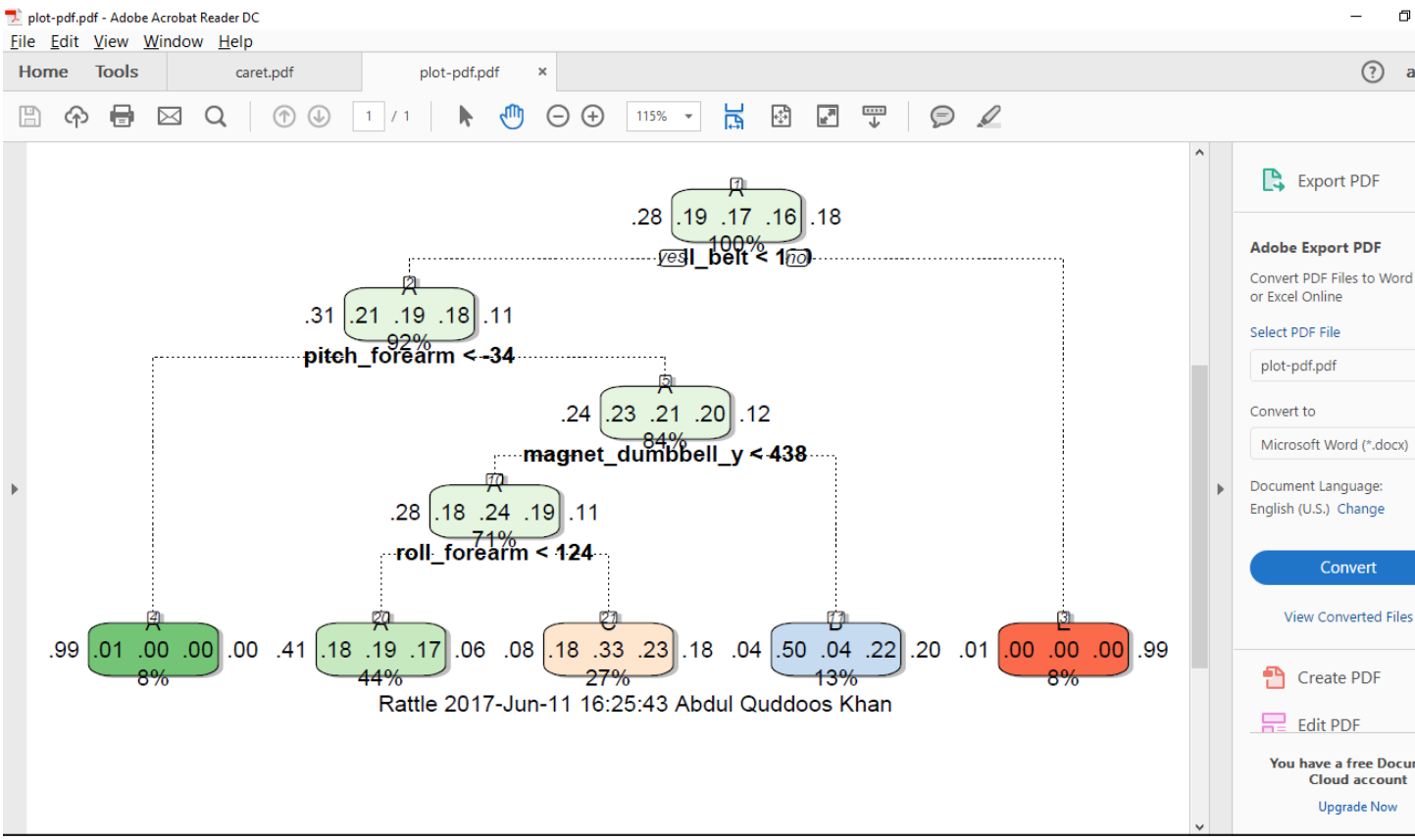
Using 5-fold cross validation

```
>control <- trainControl(method = "cv", number = 5)
>fit_rpart <- train(classe ~ ., data = train, method = "rpart",
                   trControl = control)
>print(fit_rpart, digits = 4)
CART
13737 samples
   52 predictor
    5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 10990, 10990, 10990, 10988, 10990
Resampling results across tuning parameters:

  cp       Accuracy  Kappa
  0.03723  0.5019    0.34929
  0.05954  0.4171    0.21081
  0.11423  0.3481    0.09749

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.03723.
fancyRpartPlot(fit_rpart$finalModel)
```



Now create prediction using the validation set

```
#get the predicton
predict_rpart <- predict(fit_rpart, valid)
```

```
# Show result
> (confusion_matpart <- confusionMatrix(valid$classe, predict_rpart));


Confusion Matrix and Statistics


          Reference
Prediction    A    B    C    D    E
         A 1544   21  107    0    2
         B  492  391  256    0    0
         C  474   38  514    0    0
         D  436  175  353    0    0
```

```
         E  155  138  293    0  496

Overall Statistics

               Accuracy : 0.5004
                 95% CI : (0.4876, 0.5133)
    No Information Rate : 0.5269
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3464
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.4979  0.51245  0.33749       NA  0.99598
Specificity            0.9533  0.85396  0.88262   0.8362  0.89122
Pos Pred Value         0.9223  0.34328  0.50097       NA  0.45841
Neg Pred Value         0.6303  0.92162  0.79234       NA  0.99958
Prevalence             0.5269  0.12965  0.25879   0.0000  0.08462
Detection Rate         0.2624  0.06644  0.08734   0.0000  0.08428
Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
Balanced Accuracy      0.7256  0.68321  0.61006       NA  0.94360
```

```
> (accuracy_rpart <- confusion_matpart$overall[1]);
 Accuracy
0.5004248
```

Accuracy rate is .5 and out of sample error rate is 0.5

classification tree is not predicting classes well enough.

.

Random forests

Prediction using random forest method

```
>fit_rf <- train(classe ~ ., data = train, method = "rf",
                  trControl = control)
print(fit_rf, digits = 4)
> print(fit_rf, digits = 4);
Random Forest

13737 samples
   52 predictor
    5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 10990, 10990, 10990, 10988, 10990
Resampling results across tuning parameters:

  mtry  Accuracy  Kappa
   2    0.9908    0.9884
  27    0.9906    0.9881
  52    0.9859    0.9821

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
```

Using the Validation set for RF predict result

```
> predict_rf <- predict(fit_rf, valid)
> (conf_rf <- confusionMatrix(valid$classe, predict_rf));
Confusion Matrix and Statistics

          Reference
```

```
Prediction    A    B    C    D    E
         A 1669    3    0    0    2
         B    8 1129    2    0    0
         C    0    3 1016    7    0
         D    0    1   16  944    3
         E    0    2    1    4 1075

Overall Statistics

               Accuracy : 0.9912
                 95% CI : (0.9884, 0.9934)
    No Information Rate : 0.285
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9888
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9952   0.9921   0.9816   0.9885   0.9954
Specificity            0.9988   0.9979   0.9979   0.9959   0.9985
Pos Pred Value         0.9970   0.9912   0.9903   0.9793   0.9935
Neg Pred Value         0.9981   0.9981   0.9961   0.9978   0.9990
Prevalence             0.2850   0.1934   0.1759   0.1623   0.1835
Detection Rate         0.2836   0.1918   0.1726   0.1604   0.1827
Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
Balanced Accuracy      0.9970   0.9950   0.9898   0.9922   0.9970
```

> (accuracy_rf <- conf_rf$overall[1]);

Accuracy

0.991164

As random forest is giving more accurate result and out of sample error is .009.

Although in terms of performance it took much longer to evaluate.

Final Predictions using RANDOM FOREST

We now use random forests to predict the outcome variable `classe` for the testing set.

```
> (predict(fit_rf, test_data));
 [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
```