Prediction Assignment Writeup

Bala Pelluru

03/15/2016

Read the data

- Read both training and testing instances.
- The function LOAD is to load the packages that I will use later.

```
setwd("C:\\coursera\\DataMining\\8 Practical Machine Learning\\Project")

load_packages <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}

required_packages <- c("data.table", "caret", "randomForest", "foreach", "rpart", "rpart.
plot", "corrplot")
load_packages(required_packages)
```

```
## Loading required package: data.table
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
## Loading required package: 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
##
## Loading required package: foreach
## Loading required package: rpart
## Loading required package: rpart.plot
## Loading required package: corrplot
```

```
##   data.table        caret randomForest      foreach        rpart
##         TRUE         TRUE         TRUE         TRUE         TRUE
##   rpart.plot     corrplot
##         TRUE         TRUE
```

```
training_data <- read.csv("pml_training.csv", na.strings=c("#DIV/0!"," ", "", "NA", "NA
s", "NULL"))
testing_data <- read.csv("pml_testing.csv", na.strings=c("#DIV/0!"," ", "", "NA", "NAs",
"NULL"))
```

Cleaning the data

- I need to drop columns with NAs, drop highly correlated variables and drop variables with 0 (or approx to 0) variance.
- The results are hidden as they take a very long space.

```
str(training_data)
cleantraining <- training_data[, -which(names(training_data) %in% c("X", "user_name", "ra
w_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_window", "num_windo
w"))]
cleantraining = cleantraining[, colSums(is.na(cleantraining)) == 0] #this drops columns w
ith NAs
zerovariance =nearZeroVar(cleantraining[sapply(cleantraining, is.numeric)], saveMetrics=T
RUE)
cleantraining = cleantraining[, zerovariance[, 'nzv'] == 0] #to remove 0 or near to 0 var
iance variables
correlationmatrix <- cor(na.omit(cleantraining[sapply(cleantraining, is.numeric)]))
dim(correlationmatrix)
correlationmatrixdegreesoffreedom <- expand.grid(row = 1:52, col = 1:52)
correlationmatrixdegreesoffreedom$correlation <- as.vector(correlationmatrix) #this retur
ns the correlation matrix in matrix format
removehighcorrelation <- findCorrelation(correlationmatrix, cutoff = .7, verbose = TRUE)
cleantraining <- cleantraining[, -removehighcorrelation] #this removes highly correlated
variables (in psychometric theory .7+ correlation is a high correlation)

for(i in c(8:ncol(cleantraining)-1)) {cleantraining[,i] = as.numeric(as.character(cleantr
aining[,i]))}

for(i in c(8:ncol(testing_data)-1)) {testing_data[,i] = as.numeric(as.character(testing_d
ata[,i]))} #Some columns were blank, hence are dropped. I will use a set that only includ
es complete columns. I also remove user name, timestamps and windows to have a light data
set.

featureset <- colnames(cleantraining[colSums(is.na(cleantraining)) == 0])[-(1:7)]
modeldata <- cleantraining[featureset]
featureset #now we have the model data built from our feature set.
```

Create Model

- Need to split the training data(sample) in two parts (based on best practices : 60% for training and 40% for testing is the usual).

```
idx <- createDataPartition(modeldata$classe, p=0.6, list=FALSE )
training <- modeldata[idx,]
testing <- modeldata[-idx,]
```

- A predictive model is fitted using Random Forest algorithm. Highly correlated variables were already removed but still this algorithm is robust to correlated covariates and outliers.
- A 10 fold cross validation is used.

```
control <- trainControl(method="cv", 10)
model <- train(classe ~ ., data=training, method="rf", trControl=control, ntree=250)
model
```

```
## Random Forest
##
## 11776 samples
##    23 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10598, 10598, 10599, 10598, 10598, 10599, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9710428  0.9633420  0.006397242  0.008109052
##   12    0.9696841  0.9616311  0.006697812  0.008483195
##   23    0.9637399  0.9541143  0.007434098  0.009415283
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

- The performance of the model is estimated on the validation data set.

```
predict <- predict(model, testing)
confusionMatrix(testing$classe, predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    1    0    1    0
##          B   42 1461    8    4    3
##          C    0   20 1338    8    2
##          D    2    0   56 1220    8
##          E    1    7    0   17 1417
##
## Overall Statistics
##
##                Accuracy : 0.9771
##                  95% CI : (0.9735, 0.9803)
##     No Information Rate : 0.29
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.971
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9802   0.9812   0.9544   0.9760   0.9909
## Specificity            0.9996   0.9910   0.9953   0.9900   0.9961
## Pos Pred Value         0.9991   0.9625   0.9781   0.9487   0.9827
## Neg Pred Value         0.9920   0.9956   0.9901   0.9954   0.9980
## Prevalence             0.2900   0.1898   0.1787   0.1593   0.1823
## Detection Rate         0.2842   0.1862   0.1705   0.1555   0.1806
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9899   0.9861   0.9748   0.9830   0.9935
```

```
accuracy <- postResample(predict, testing$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9770584 0.9709611
```

- The estimated accuracy of the model is 97.6% and the estimated out of sample error is 2.4%.

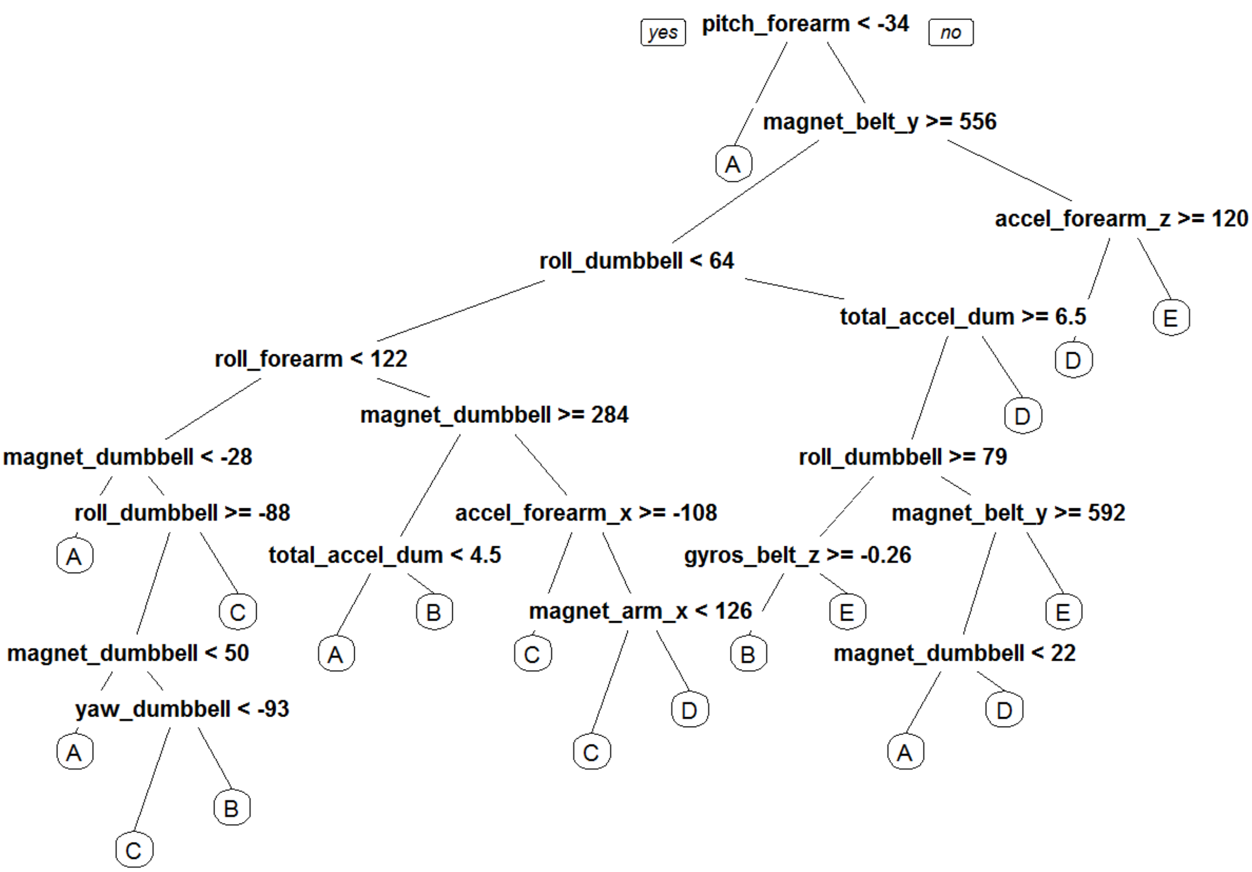
Predictions

- The model is aplied to the original testing data.

```
result <- predict(model, training[, -length(names(training))])
result
```

Tree

```
treeModel <- rpart(classe ~ ., data=cleantraining, method="class")
prp(treeModel)
```



Predictions

```
write_predictions = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

testing_data <- testing_data[featureset[featureset!='classe']]
Predictions <- predict(model, newdata=testing_data)
Predictions
```

```
##  [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
```

```
write_predictions(Predictions)
```