Machine Learning

GB

Monday, May 15, 2015

Summary

The goal of the project is to built the model that predicts the manner in which the exercise was done and data that was used. The report describes and details the steps how the model was built, detailing the used cross validation model and thinking behind the expected out of sample error is, and variables choosen that were used in the model.

Out-Sample-Error

In statistical data modeling, statistical tests of a model's forecast performance are commonly conducted by splitting a given data set into an in-sample period, which is normally called Training Data set and is used for the initial parameter estimation and model selection.

The second split normally called the testing data set is used to evaluate forecasting performance, based on the model built using our Training Data set. It's this out-of-sample data that will be the basis of calculating our accuracy or perfomacy errors, since empirical evidence based on out-of-sample forecast performance is generally considered more trustworthy than evidence based on in-sample performance, which can be more sensitive to outliers and data mining. Out-of-sample forecasts also better reflect the information available to the forecaster in "real time".

Preparing

```
library(caret)
```

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

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

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

```
library(ggplot2)

training<-read.csv("pml-training.csv",na.strings=c("NA",""))
testing<-read.csv("pml-testing.csv",na.strings=c("NA",""))
```

Data Exploring

```
dim(training)
```

```
## [1] 19622   160
```

```
unique(training$classe)
```

```
## [1] A B C D E
## Levels: A B C D E
```

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

There are 19622 observation in traning dataset, including 160 variables

Data Cleasing

Since we have a lot of variables containing NA values, these will be removed as they can distort our results and also variables that do not add value will be removed

```
NA_Count <- sapply(1:dim(training)[2],function(x)sum(is.na(training[,x])))
NA_list<- which(NA_Count>0)

training <- training[,-NA_list]
training <- training[,-c(1:7)]
training$classe = factor(training$classe)

# Similarily process the testing data set
testing <- testing[,-NA_list]
testing <- testing[,-c(1:7)]
```

Partitioning our training data into training data (trainData) and validation data (testData)

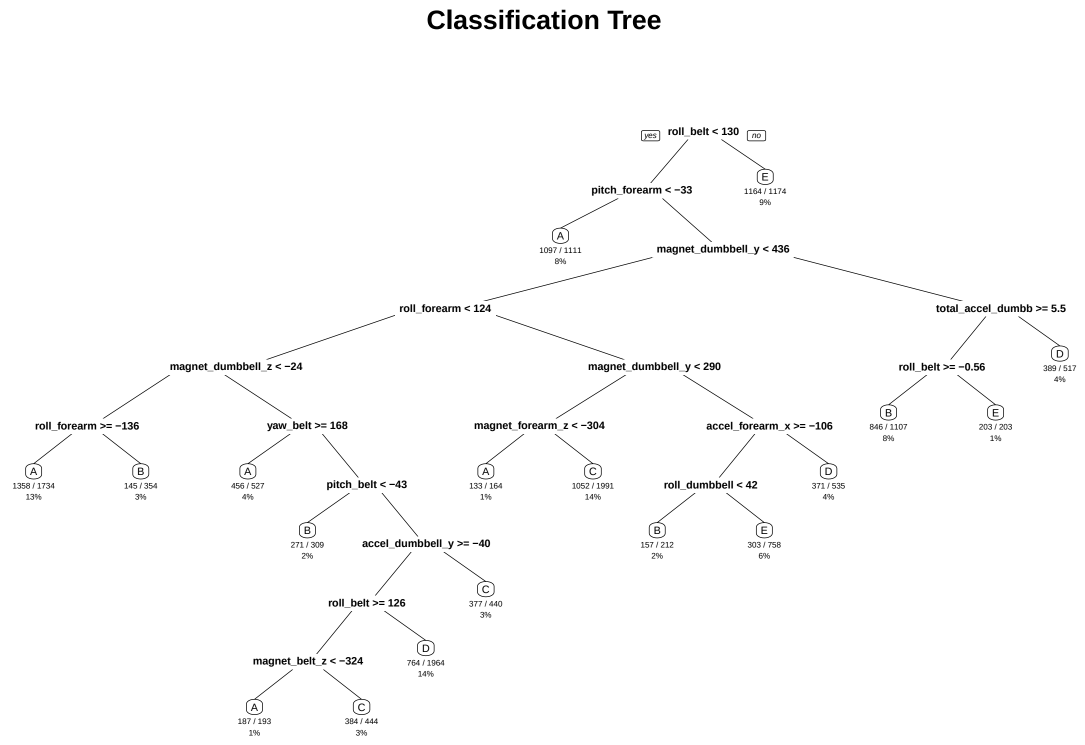
```
set.seed(1234)
inTrain <- createDataPartition(training$classe,p=0.70,list=F)
trainData <- training[inTrain,]
testData <- training[-inTrain,]
```

Two data modeling algorithms are going to be compared and the one with the best accuracy will be used for prediction on the testing data and calculating out-of-sample error. The two algorithms in considering are:

- Classification Decision Tree (rpart)
- Random Forests (rf from caret)

Decision Tree Data Modeling

```r
modTreeFit <- rpart(classe ~ ., data = trainData, method="class")
predictionClass <- predict(modTreeFit, testData, type = "class")
rpart.plot(modTreeFit, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



```r
confusionMatrix(predictionClass, testData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1364  169   24   48   16
##          B   60  581   46   79   74
##          C   52  137  765  129  145
##          D  183  194  125  650  159
##          E   15   58   66   58  688
##
## Overall Statistics
##
##                Accuracy : 0.6879
##                  95% CI : (0.6758, 0.6997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6066
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8148  0.51010   0.7456   0.6743   0.6359
## Specificity            0.9390  0.94543   0.9047   0.8657   0.9590
## Pos Pred Value         0.8415  0.69167   0.6230   0.4958   0.7774
## Neg Pred Value         0.9273  0.88940   0.9440   0.9314   0.9212
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2318  0.09873   0.1300   0.1105   0.1169
## Detection Prevalence   0.2754  0.14274   0.2087   0.2228   0.1504
## Balanced Accuracy      0.8769  0.72776   0.8252   0.7700   0.7974
```

Cross Validation Data Modeling

```
cv3 <- trainControl(method="cv",number=5)
modrf <- train(classe~., data=trainData, method="rf",trControl=cv3,ntree=250)
modrf
```

```
## 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: 10989, 10989, 10990, 10989, 10991
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9909005  0.9884883  0.001091905  0.001382208
##   27    0.9916283  0.9894086  0.002650396  0.003353723
##   52    0.9887166  0.9857252  0.003184175  0.004029297
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Predict and calculate out-of-sample error on the validation data set.

```
predictRf <- predict(modrf, testData)
confusionMatrix(testData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##           A 1674    0    0    0    0
##           B   11 1127    1    0    0
##           C    0    3 1019    4    0
##           D    0    2    5  956    1
##           E    0    1    3    3 1075
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9942          
##                  95% CI : (0.9919, 0.996) 
##     No Information Rate : 0.2863          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9927          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9935   0.9947   0.9912   0.9927   0.9991
## Specificity            1.0000   0.9975   0.9986   0.9984   0.9985
## Pos Pred Value         1.0000   0.9895   0.9932   0.9917   0.9935
## Neg Pred Value         0.9974   0.9987   0.9981   0.9986   0.9998
## Prevalence             0.2863   0.1925   0.1747   0.1636   0.1828
## Detection Rate         0.2845   0.1915   0.1732   0.1624   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9967   0.9961   0.9949   0.9956   0.9988
```

```r
# postResample Calculates performance across resamples
accuracy <- postResample(predictRf, testData$classe)
accuracy
```

```
##  Accuracy     Kappa 
## 0.9942226 0.9926907
```

From above output from the Confusion Matrix outputs for both the Decision Tree and the Random Forests Modeling, it's noted that the Random Forests algorithm has a better accuracy 99.42% compared to 68.79% for Decision Tree algorithm, hence the rest of the report will be based on the Random Forests algorithm data modeling.

Out-of-sample error

```r
# 1 minus the model accuracy
oose <- 1 - as.numeric(confusionMatrix(testData$classe, predictRf)$overall[1])
oose
```

```
## [1] 0.0057774
```

So, the estimated accuracy of the model is **99.42%** and the estimated out-of-sample error is **0.58%**.

Predict on the Testing Data Set

Now, applying the model to the original testing data set downloaded from the data source and cleansed from above.

```
result <- predict(modrf, testing)
result
```

```
##  [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
```

Prediction Assignment Submission

The function creates the files to apply the machine learning algorithm built to each of the 20 test cases in the testing data set.

```
pml_write_files = 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)
  }
}
pml_write_files(result)
```

Conclusion

The random forest appraoch with cross validation proved to be extremely accurate than the decision tree approach. In this analysis one part that was not considered was how differet predictor correlations could affect the predictions in this particular analysis.