Prediction Assignment Writeup

Dilip Suwal

September 24, 2017

Machine Learning: Prediction Assignment Writeup

Introduction

This document summarizes the work done for the Prediction Assignment Writeup project for the Coursera Practical Machine Learning course. It decribes the machine learning methods to predict the manner in which participants did exercise.The data for this project involves readings from wearable fitness trackers such as Jawbone Up, Nike FuelBand, and Fitbit.

Data

The training data for this project are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Both training and test data have already been dowloaded to my machine at the following location: C:\WorkSpace\DataScience\8.MachineLearning\W4\Assignment

Environment Setup

```
setwd("C:/WorkSpace/DataScience/8.MachineLearning/W4/Assignment")
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2
```

Loading Data

```
TrainingSet <- read.csv("pml-training.csv", na.strings=c("NA",""),
strip.white=T)
TestingSet <- read.csv("pml-testing.csv", na.strings=c("NA",""),
strip.white=T)
```

Data Cleansing

Now, let us remove all columns that contains NA and remove features that are not in the testing dataset. Since the testing dataset has no time-dependence, these values are useless

and can be disregarded. Cosidering this, let us remove the first 7 features since they are related to the time-series.

```
features <- names(TestingSet[,colSums(is.na(TestingSet)) == 0])[8:59]

# Only use features used in testing cases.
TrainingSet <- TrainingSet[,c(features,"classe")]
TestingSet <- TestingSet[,c(features,"problem_id")]
dim(TrainingSet); dim(TestingSet);

## [1] 19622    53

## [1] 20 53
```

Partitioning the Data Set

Let us now split data into a training data set and a testing data set with the ratio of 60 to 40. This will allow us to estimate the out of sample error of our predictor.

```
inTrain <- createDataPartition(TrainingSet$classe, p=0.6, list=FALSE)
training <- TrainingSet[inTrain,]
testing <- TrainingSet[-inTrain,]
dim(training); dim(testing);

## [1] 11776    53

## [1] 7846   53
```

Building Model

In order to predict with high accuracy about the manner in which participants did exercise, we will evalute the following five different algorithms

1. Random Forest (RF)
2. Classification and Regression Trees (CART)
3. Linear Discriminant Analysis (LDA)
4. k-Nearest Neighbors (kNN)
5. Support Vector Machines (SVM) with a linear kernel

Before building models, let us prepare the cross validation with 5 fold. This will split our data set into 5 parts, train in 4 and test on 1.

```
control <- trainControl(method="cv", number=5)
metric <- "Accuracy"
```

Now, let us build five models as stated above.

```
#Random Forest
set.seed(777)
fit.rf <- train(classe~., data=training, method="rf", metric=metric,
trControl=control)
```

```
#Classification and Regression Trees
set.seed(777)
fit.cart <- train(classe~., data=training, method="rpart", metric=metric,
trControl=control)

#Linear Discriminant Analysis
set.seed(777)
fit.lda <- train(classe~., data=training, method="lda", metric=metric,
trControl=control)

#k-Nearest Neighbors
set.seed(777)
fit.knn <- train(classe~., data=training, method="knn", metric=metric,
trControl=control)

#Support Vector Machines
set.seed(777)
fit.svm <- train(classe~., data=training, method="svmRadial", metric=metric,
trControl=control)
```

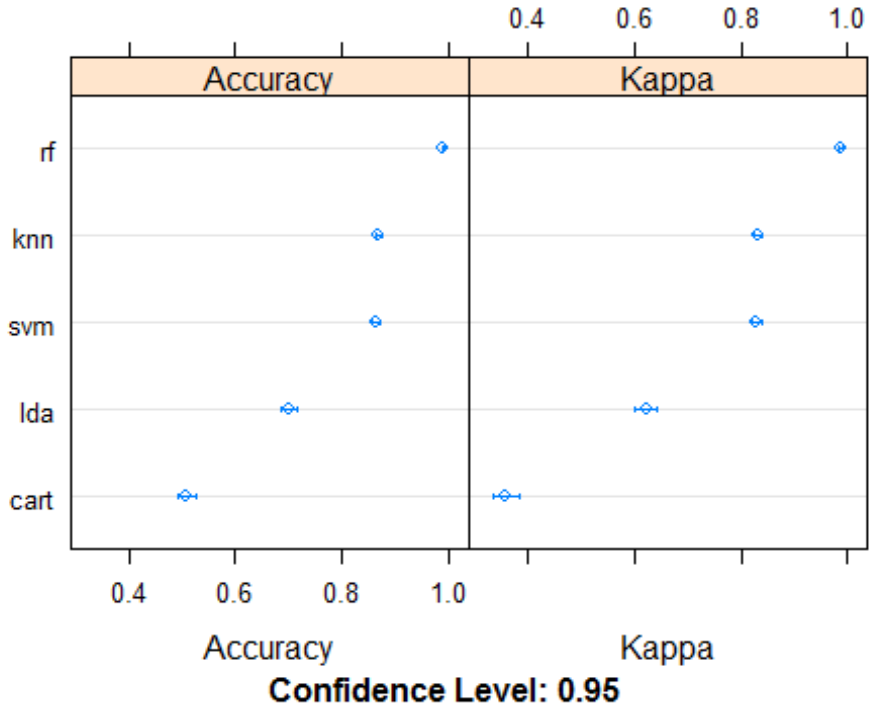
Select Best Model

```
results <- resamples(list(rf=fit.rf, cart=fit.cart, lda=fit.lda, knn=fit.knn,
svm=fit.svm))
summary(results)

##
## Call:
## summary.resamples(object = results)
##
## Models: rf, cart, lda, knn, svm
## Number of resamples: 5
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf   0.9868309 0.9885399 0.9893843 0.9895547 0.9915074 0.9915110    0
## cart 0.4898132 0.4974511 0.5174024 0.5088314 0.5176221 0.5218684    0
## lda  0.6895966 0.6918506 0.6926995 0.7007499 0.7141037 0.7154989    0
## knn  0.8616299 0.8646010 0.8704333 0.8682925 0.8717622 0.8730361    0
## svm  0.8564756 0.8599321 0.8633277 0.8642163 0.8700085 0.8713376    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf   0.9833418 0.9855019 0.9865702 0.9867868 0.9892582 0.9892618    0
## cart 0.3332726 0.3444342 0.3707364 0.3591244 0.3708935 0.3762856    0
## lda  0.6073917 0.6105559 0.6108003 0.6213345 0.6380301 0.6398947    0
## knn  0.8248647 0.8287043 0.8359713 0.8333720 0.8378017 0.8395178    0
## svm  0.8188417 0.8233505 0.8276275 0.8286497 0.8357613 0.8376678    0
```

From the above analysis summary, we can see that the most accurate model is random forest model. This we can visualize as below as well.

```
# compare accuracy of models
dotplot(results)
```



The result for random forest model can be summarized as below.

```
# compare accuracy of models
print(fit.rf)

## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9422, 9421, 9420, 9421, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9869230  0.9834541
##   27    0.9895547  0.9867868
##   52    0.9808930  0.9758275
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Make Prediction

As random forest model is the most accurate one, we will now estimate the accuracy of the model with the the testing data set. This will give us an independent final check on the accuracy of the best model. let us run the random forest model directly on the testing set and summarize the results in a confusion matrix.

```
# Estimate the accuracy of the model on testing data set
predictions <- predict(fit.rf, testing)
confusionMatrix(predictions, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2225   10    0    0    0
##          B    5 1504    8    4    1
##          C    1    4 1356   12    3
##          D    0    0    4 1270    3
##          E    1    0    0    0 1435
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9907, 0.9946)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9969   0.9908   0.9912   0.9876   0.9951
## Specificity            0.9982   0.9972   0.9969   0.9989   0.9998
## Pos Pred Value         0.9955   0.9882   0.9855   0.9945   0.9993
## Neg Pred Value         0.9988   0.9978   0.9981   0.9976   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2836   0.1917   0.1728   0.1619   0.1829
## Detection Prevalence   0.2849   0.1940   0.1754   0.1628   0.1830
## Balanced Accuracy      0.9975   0.9940   0.9941   0.9932   0.9975
```

Conclusion

From above analysis summary, we can see that the accuracy of random forest model is 98.94%