the Analysis on Classe

Lukas Chen

1. summary

This paper focused on predicting the manner in which the 6 participants did their exercise. I used some of the well-known machine learning algorithm to build the predictions, such as Decision Tree, SVM, bagging method and so on. And then I choosing one of the algorithm, and took the cross-validation to estimate the stability of the model. At last, I used three models to predict the test data set and vote the final result based on each prediction

2. getting data

```
load("D:/kuaipan/TEST/Data Science/train.RData")
load("D:/kuaipan/TEST/Data Science/test.RData")
```

3. data cleaning

```
missing_filter <- function(df.test, df.train){
      df.test.new <- df.test[colSums(is.na(df.test)) < 10]
      df.train.new <- df.train[names(df.train) %in% names(df.test.new)]
      res <- list(df.test.new, df.train.new)
      return(res)
}
res <- missing_filter(df.test = test, df.train = train)
test.new <- res[[1]]
train.new <- cbind(res[[2]], data.frame(train$classe))

# get rid of the missing data, and make sure the train set and test set have the same variab
les basicly
```

4. variable exploring

```
library(caret)
```

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(rpart.plot)
```

```
## Loading required package: rpart
```

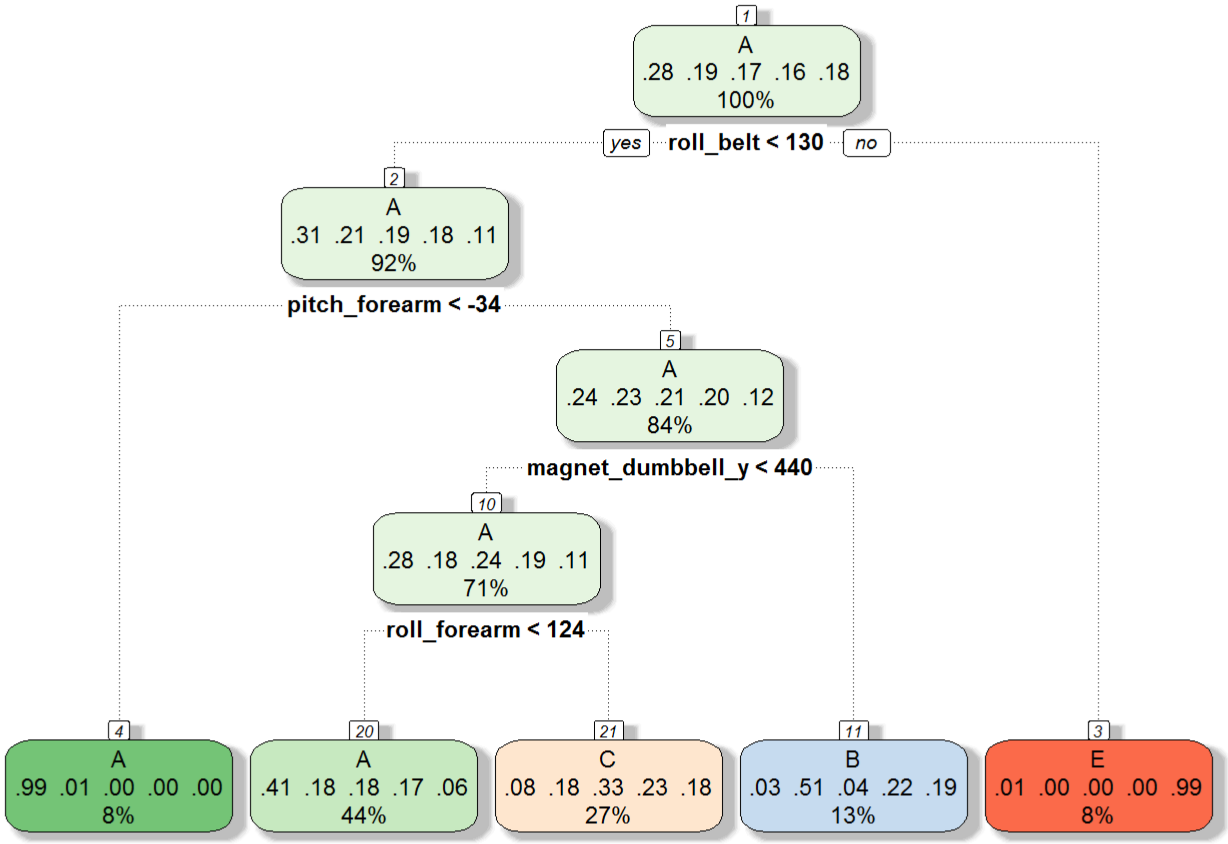
```
train.new <- train.new[,-c(3:5)];train.new <- train.new[,-c(1)]

fit.rpart <- train(train.classe ~ ., data = train.new, method = "rpart")
fit.Imp <- varImp(fit.rpart$finalModel, scale = T, surrogates = F, competes = T)

library(rattle)
```

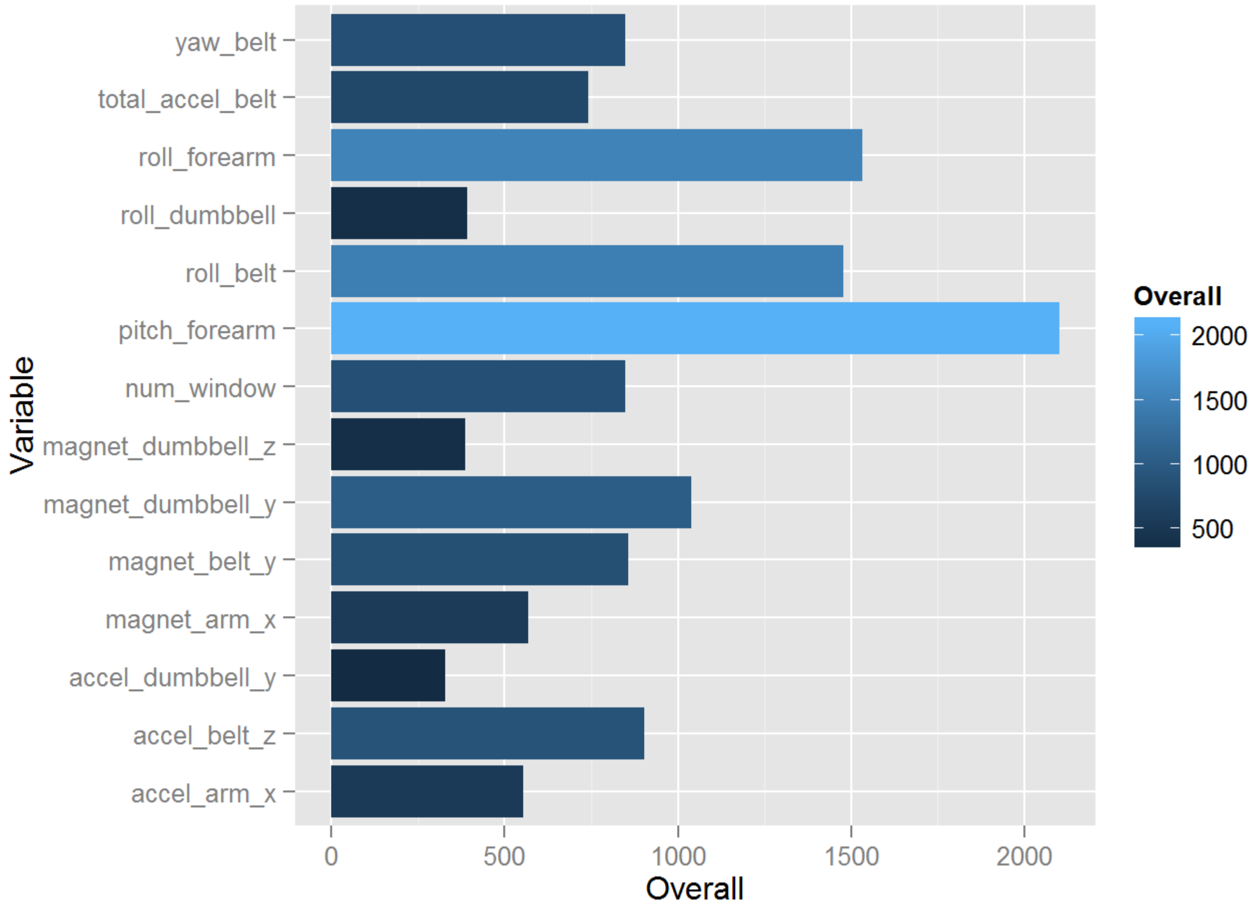
```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## XXXX 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## □□'rattle()'□□□□□□□□□□□□□□
```

```
fancyRpartPlot(fit.rpart$finalModel)
```



Rattle 2015-九月-27 08:46:55 Lukas

```
fit.Imp$Variable <- row.names(fit.Imp)
fit.Imp2 <- arrange(fit.Imp, desc(Overall))
ggplot(fit.Imp2[1:14,], aes(x = Variable, y = Overall)) + geom_bar(aes(fill = Overall), stat
= "identity") + coord_flip()
```




```
# use rpart to fit a model first, and save the important variables for choosing models
```

5. model choosing

```
train.model <- cbind(train.new[names(train.new) %in% fit.Imp[1:14,2]], train.new$train.classe
)
names(train.model)[15] <- "classe"

fit.rpart <- tryCatch(train(classe ~ ., data = train.model, method = "rpart",
                            trControl = trainControl(method = "repeatedcv", number = 10, repe
ats = 3),
                            tuneGrid = data.frame(.cp = seq(from = 0.001, to = 0.1, length =
10))), error = function(e) return(NA))
pred.rpart <- predict(fit.rpart, newdata = train.model)
err.rpart <- sum(train.model$classe == pred.rpart)/length(train.model$classe); err.rpart
```

```
## [1] 0.9394047
```

```
confusionMatrix(pred.rpart, train.model$classe)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 5453  133   17   17   16
##          B   34 3340   63   41   68
##          C   32  168 3264  119   47
##          D   30  101   66 3012  112
##          E   31   55   12   27 3364
##
## Overall Statistics
##
##                Accuracy : 0.9394
##                  95% CI : (0.936, 0.9427)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9234
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9772   0.8796   0.9538   0.9366   0.9326
## Specificity            0.9870   0.9870   0.9774   0.9812   0.9922
## Pos Pred Value         0.9675   0.9419   0.8992   0.9070   0.9642
## Neg Pred Value         0.9909   0.9716   0.9901   0.9875   0.9849
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2779   0.1702   0.1663   0.1535   0.1714
## Detection Prevalence   0.2872   0.1807   0.1850   0.1692   0.1778
## Balanced Accuracy      0.9821   0.9333   0.9656   0.9589   0.9624
```

```r
library(ipred)
fit.bagging <- tryCatch(bagging(classe ~ ., data = train.model), error = function(e) return(N
A))
pred.bagging <- predict(fit.bagging, newdata = train.model)
err.bagging <- sum(train.model$classe == pred.bagging)/length(train.model$classe); err.bagging
```

```
## [1] 0.999949
```

```r
confusionMatrix(pred.bagging, train.model$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3796    0    0    0
##          C    0    1 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9999
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9999
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9997   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   0.9999   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9997   1.0000   1.0000
## Neg Pred Value         1.0000   0.9999   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   0.9999   1.0000   1.0000   1.0000
```

```
library(e1071)
fit.svm <- tryCatch(svm(classe ~ ., data = train.model), error = function(e) return(NA))
pred.svm <- predict(fit.svm, newdata = train.model)
err.svm <- sum(train.model$classe == pred.svm)/length(train.model$classe); err.svm
```

```
## [1] 0.8861992
```

```
confusionMatrix(pred.svm, train.model$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5440  327   16    9    8
##          B   47 2818  131   62   64
##          C   52  455 3159  418  126
##          D   31  179   76 2687  124
##          E   10   18   40   40 3285
##
## Overall Statistics
##
##                Accuracy : 0.8862
##                  95% CI : (0.8817, 0.8906)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.856
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9749   0.7422   0.9231   0.8355   0.9107
## Specificity            0.9744   0.9808   0.9351   0.9750   0.9933
## Pos Pred Value         0.9379   0.9026   0.7504   0.8676   0.9682
## Neg Pred Value         0.9899   0.9407   0.9829   0.9680   0.9802
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2772   0.1436   0.1610   0.1369   0.1674
## Detection Prevalence   0.2956   0.1591   0.2146   0.1578   0.1729
## Balanced Accuracy      0.9746   0.8615   0.9291   0.9053   0.9520
```

```
# use rpart, bagging and svm to build the model on the train set, and we can see that baggin
g has the highest accuracy, but maybe overfitting
```

6. cross-validation

```
cross_validation <- function(df, cross = 10){
  if(!is.data.frame(df)){
    stop("The data isn't formatted as dataframe!")
  }
  if(cross == 1){
    stop("The cross could not be 1!")
  }
  if(nrow(df)/cross < 30){
    print("The result may not be vary specific!")
  }

  index.1 <- 1:nrow(df)
  set.seed(1)
  index.2 <- sample(x = index.1, size = nrow(df))
  index.3 <- rep(x = 1:cross, time = ceiling(nrow(df)/cross))[index.1]

  err.train <- rep(0,cross)
  err.test <- rep(0,cross)

  for(i in 1:cross){
    test.index <- index.1[index.3 == i]
    train <- df[-test.index, ]
    test <- df[test.index, ]

    fit <- tryCatch(svm(classe ~ ., data = train.model), error = function(e) return(NA))
    pred.train <- predict(fit, newdata = train)
    pred.test <- predict(fit, newdata = test)

    err.train[i] <- sum(train$classe == pred.train)/length(train$classe)
    err.test[i] <- sum(test$classe == pred.test)/length(test$classe)
  }
  res <- list(err.train = err.train, err.test = err.test)
  return(res)
}
cv <- cross_validation(df = train.model, cross = 10)
```

```
cv$err.train
```

```
##  [1] 0.8859505 0.8861770 0.8859570 0.8859003 0.8864100 0.8863533 0.8859570
##  [8] 0.8868063 0.8864666 0.8860136
```

```
mean(cv$err.train)
```

```
## [1] 0.8861992
```

```
cv$err.test
```

```
##  [1] 0.8884361 0.8863984 0.8883792 0.8888889 0.8843017 0.8848114 0.8883792
##  [8] 0.8807339 0.8837920 0.8878695
```

```
mean(cv$err.test)
```

```
## [1] 0.886199
```

```
# take cross_validation with svm to estimate the stability of the model
```

7. predict on test set

```
pred.test1 <- predict(fit.rpart, newdata = test.new)
pred.test1
```

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

```
pred.test2 <- predict(fit.bagging, newdata = test.new)
pred.test2
```

```
##  [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
```

```
pred.test2 <- predict(fit.svm, newdata = test.new)
pred.test2
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  C  A  C  A  A  C  D  D  A  A  B  C  B  A  E  E  A  D  B  B
## Levels: A B C D E
```

```
res <- data.frame(pred.rpart = pred.test1, pred.bagging = pred.test2, pred.svm = pred.test2)
```

```
pred.final <- c()
for(i in 1:nrow(res)){
      if(res[i,1] == res[i,2] | res[i,1] == res[i,3]){
          pred.final[i] <- res[i,1]
      }else{
          pred.final[i] <- res[i,2]
      }
}
res$pred.final <- pred.final
res$final[res$pred.final == 1] <- "A"
res$final[res$pred.final == 2] <- "B"
res$final[res$pred.final == 3] <- "C"
res$final[res$pred.final == 4] <- "D"
res$final[res$pred.final == 5] <- "E"
res
```

```
##    pred.rpart pred.bagging pred.svm pred.final final
## 1           B            C        C          3     C
## 2           A            A        A          1     A
## 3           B            C        C          3     C
## 4           A            A        A          1     A
## 5           A            A        A          1     A
## 6           E            C        C          3     C
## 7           D            D        D          4     D
## 8           D            D        D          4     D
## 9           A            A        A          1     A
## 10          A            A        A          1     A
## 11          C            B        B          2     B
## 12          C            C        C          3     C
## 13          B            B        B          2     B
## 14          A            A        A          1     A
## 15          E            E        E          5     E
## 16          E            E        E          5     E
## 17          A            A        A          1     A
## 18          B            D        D          4     D
## 19          B            B        B          2     B
## 20          B            B        B          2     B
```

```
# used three models to predict the test data set and vote the final result based on each pre
diction
```