Exercise Manner Prediction

Hugo Barros

November 19, 2017

Summary

The goal of this assignment is to predict the manner in which an exercise was performed.

We will perform some data cleaning, variable selection, model creation and predictions.

Our model of choice for prediction will be the Random Forrest. It is generally, and also in this case, more accurate.

Data Cleaning

We have a training and a test dataset. So our first task will be to read the two data sets(treating empty cells as NAs), and join them, without the last column of each (because they are different and also because they're not meant to go through data cleaning) into one dataframe.

```
traindat <- read.csv("c:/Users/Dafh/Documents/R/Work/Machine Learning/trainDat.csv", na.strings = c("",

testdat <- read.csv("c:/Users/Dafh/Documents/R/Work/Machine Learning/testDat.csv", na.strings = c("",NA

joint <- rbind(traindat[,-160], testdat[,-160])
```

Now we will remove columns with only zero-values, that are highly correlated, and those where 90% or more of their values are NAs.

```
pre_joint <- preProcess(joint,method = c("zv","corr"))
joint1 <- predict(pre_joint,joint)
na_count <- sapply(joint1, function(y) sum(is.na(y)))
zeroNA <- joint1[,(na_count/nrow(joint1)) < 0.90]
```

Now we have a dataframe without columns composed of only zero-values, or mostly of NAs, which makes us ready to split the dataframe back into a training and a test datasets. We will also add the last column of training data(classe) that we removed when we joined the two datasets

```
trClean <- zeroNA[1:19622,]
testClean <- zeroNA[19623:19642,]
trClean$classe <- traindat$classe
```

Variable Selection

On the train dataframe, it is imperative to do some variable selection in order to get rid of some of the noise of variables that are not likely to be related to the outcome. Some of these variables are: X, user_name, the timestamps(all three of them).

```
trClean <- trClean[,-c(1:6)]
```

We consider our training set clean enough for a fair shot at modeling and accurate predictions.

Modeling (and train set accuracy)

Three types of models will be created and applied on the train set and then on the test set: lda, rpart, and radomForrest.

The first two models will be just for demonstration purposes. We will see, by their accuracy on the train set and then predictions on test set, that their outcome prediction is not likely accurate enough.

We will use random forrest's prediction as our pick for the test set and quiz questions.

LDA

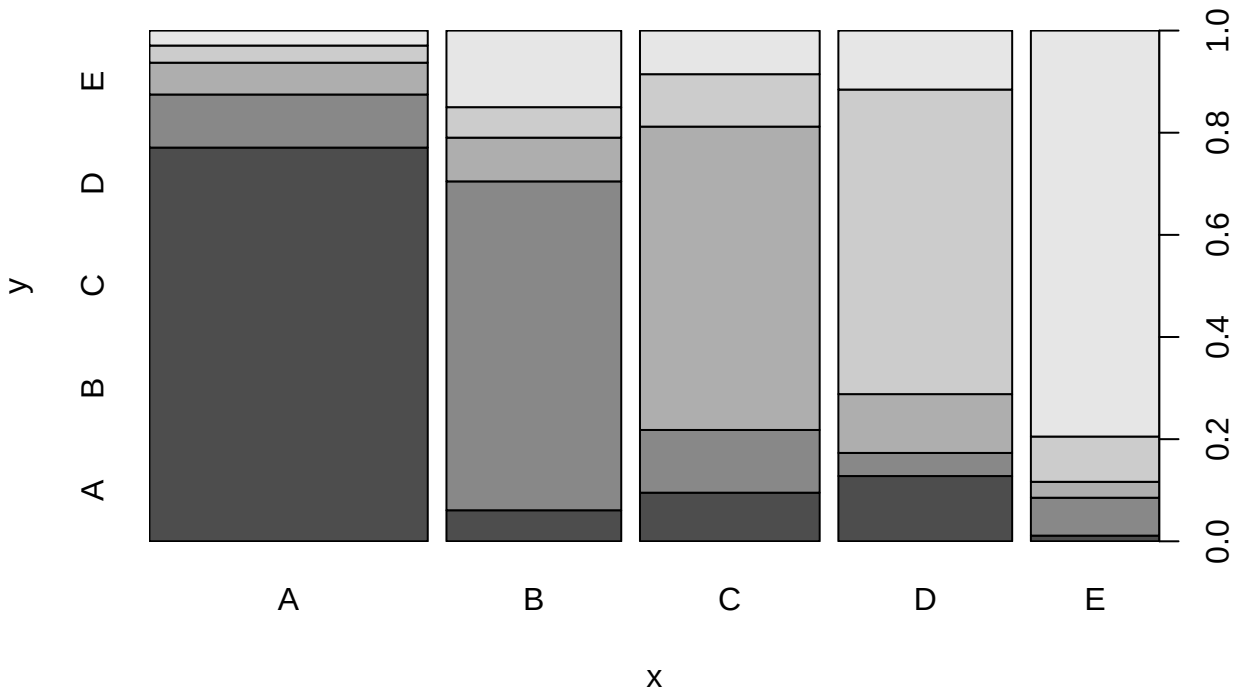
(lda modeling, prediction on train data and then test data)

```
ldamod <- train(classe ~ ., data = trClean, method="lda")

confusionMatrix(predict(ldamod,trClean),trClean[,46])$overall[1]
```

```
##  Accuracy
## 0.6840281
```

```
plot(predict(ldamod,trClean),trClean[,46])
```



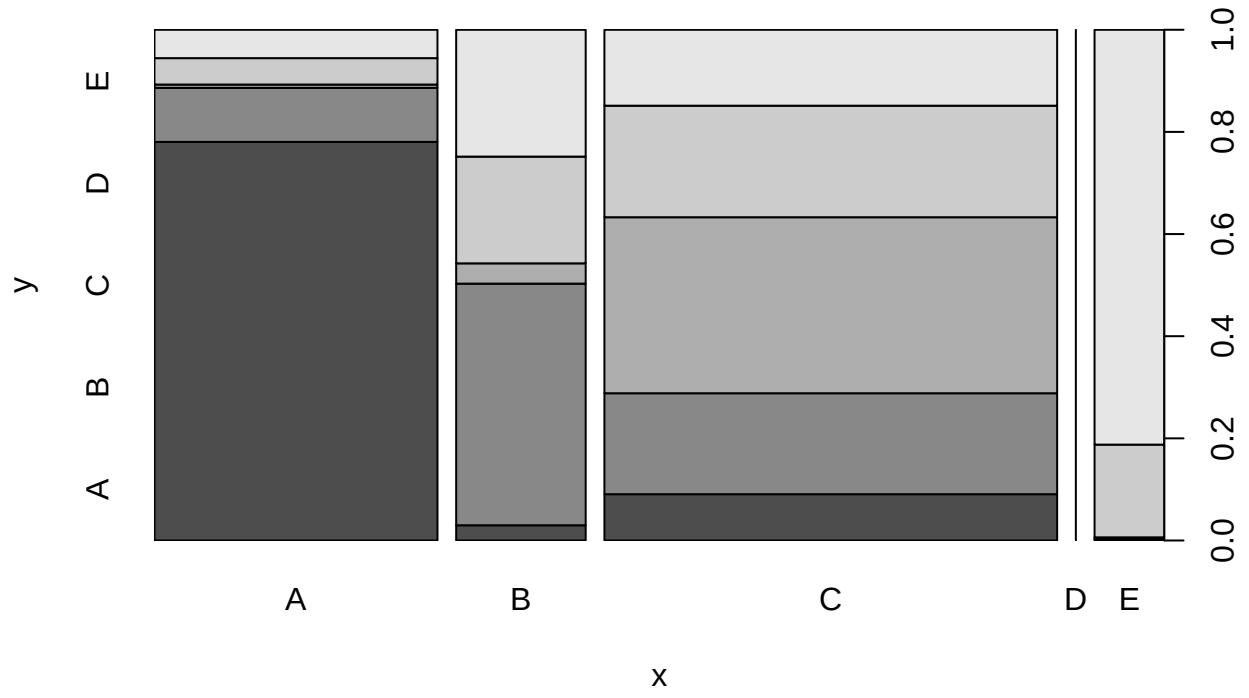
RPART

(rpart modeling, prediction on train data and then test data)

```
rpartmod <- train(classe ~ ., data = trClean, method="rpart")
confusionMatrix(predict(rpartmod,trClean),trClean[,"classe"])$overall[1]
```

```
##  Accuracy
## 0.5292529
```

```
plot(predict(rpartmod,trClean),trClean[,46])
```



As far as accuracy on the train data, lda is performing better than rpart. Still, both models are not very accurate even on the data where they were trained. However, that does not mean they wouldn't perform better on the test data.

RandomForrest

we will apply a couple of tweaks to randomForrest in order to make it less computationally demanding. One of them will be for parallel execution/processing and the other is the number of folds used in cross-validation.

Cross-validation will be used with 10-folds since it seems to be not too expensive computationally-wise, while also allowing the model to still be accurate enough.

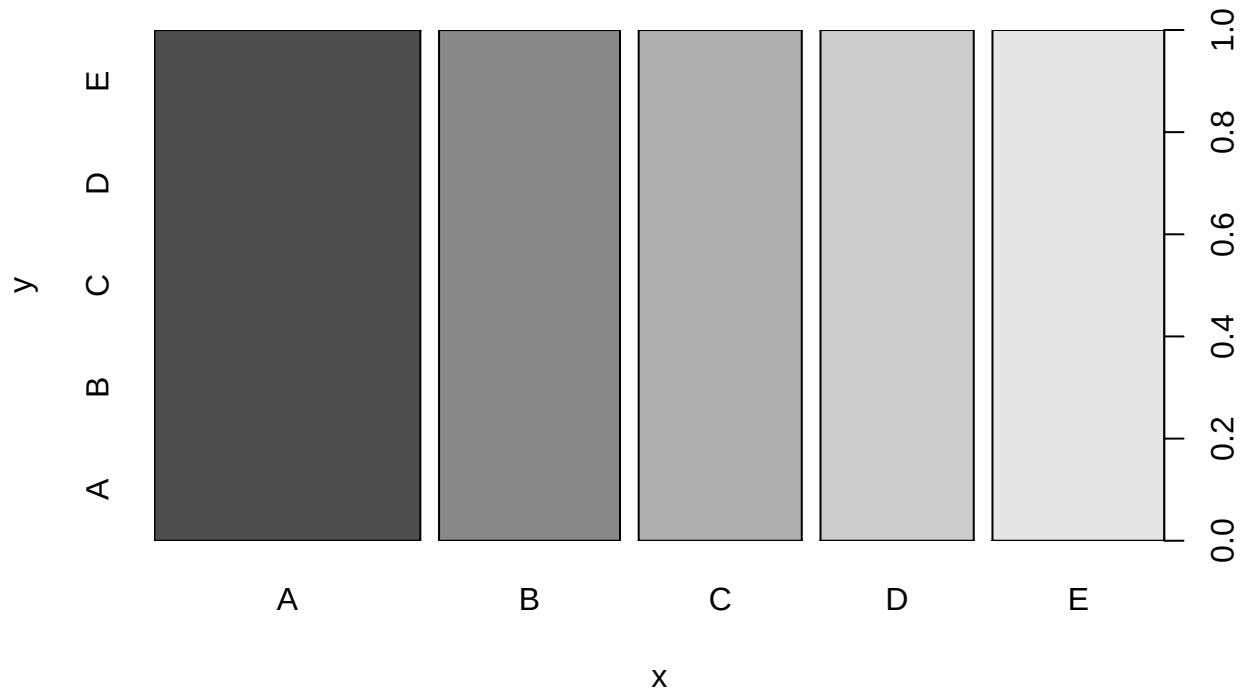
```
my_cluster <- makeCluster(detectCores()-1)

registerDoParallel(my_cluster)
fitControl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
x <- trClean[,-46]
y <- trClean[,46]
fit <- train(x,y,method = "rf", data=trClean,trControl = fitControl)
```

```
confusionMatrix(predict(fit,trClean),trClean[,"classe"])$overall[1]
```

```
## Accuracy
##        1
```

```
plot(predict(fit,trClean),trClean[,46])
```



The expected out-of-sample error of our random forrest model is 0.12% () (as per our model's "final model" parameter). It seems to us, that the random forrest is quite accurate .

```
fit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, data = ..1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 23
##
##         OOB estimate of  error rate: 0.13%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5577    2    0    0    1 0.0005376344
## B    4 3791    2    0    0 0.0015801949
## C    0    5 3417    0    0 0.0014611338
## D    0    0    7 3208    1 0.0024875622
## E    0    0    0    3 3604 0.0008317161
```

Predictions (on test set)

We will now use all three models for prediction on the test data.

LDA:

```
  lda_test_prediction <- predict(ldamod,testClean)
  lda_test_prediction
```

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

RPART:

```
  rpart_prediction <- predict(rpartmod,testClean)
  rpart_prediction
```

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

RandomForrest

```
  rf_prediction <- predict(fit,testClean)
  rf_prediction
```

```
##  [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
```

Conclusion

Because there is still a quiz part of the assignment, we won't reveal how well our models performed (well, you can go try it for yourself ;)). However, we know,by trying our predictions, that Random Forrest was far more accurate than the other two methods.

`We hope this document was clear, explanatory, and demonstrative of the steps taken to use data to predi`

Thank You,