Practical Machine Learning Course Project

Emily Kubicek

2/27/2020

R Markdown

They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Load Data

```
# Load in training/test sets
training <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.c
sv")

testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v")
```

Explore/clean data

```
## [1] 19622   100
```

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

```
## [1] 787856
```

```
## [1] 820
```

```
## [1] 19622    59
```

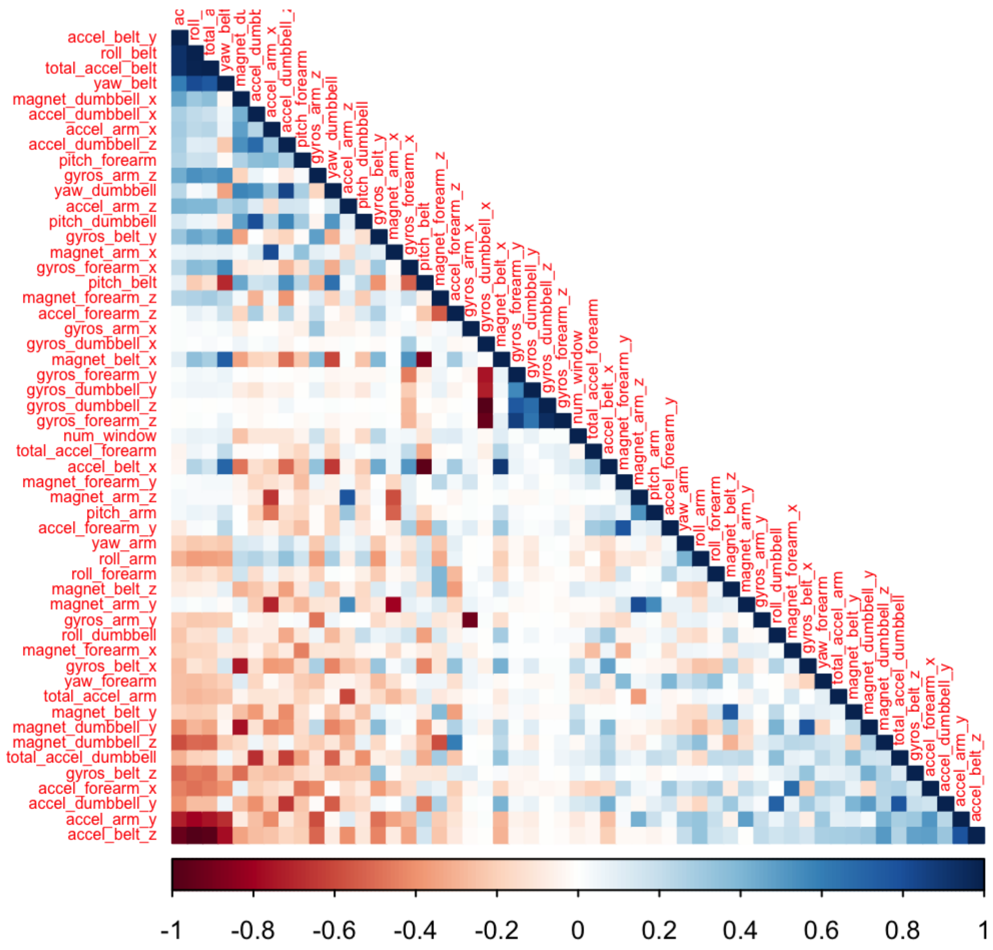
```
## [1] 19622   100
```

```
## [1] 19622    54
```

The above cleaning has changed our data significantly. We now only have 54 variables to work with for both training and testing, with all variables contianing mostly relevant (i.e. not NA data). However, in order to see the accuracies of our model, we must further break apart the training set. Leaving the inital test set for validation.

```
inTrain <- createDataPartition(training4$classe, p = 3/4, list = FALSE)
training5 <- training4[inTrain,]
testing5 <- training4[-inTrain,]
```

Conduct inital correlation analysis



The above correlation matrix gives us an overview of our variables. We can see that many of our variables are highly correlated (represented by dark colors in the matrix). Given the high amount of variables, PCA could be performed to create powerful components. However, for this analysis we will be attempting to fit multiple models instead to which performs best.

Fit models

Random Forest

```
# fit cleaned data to random forest and find accuracy
set.seed(11111)
rfmod <- randomForest(classe ~ ., data = training5)
rfmodpred <- predict(rfmod, testing5)
confusionMatrix(rfmodpred, testing5$classe)$overall[1]
```

```
##  Accuracy
## 0.9977569
```

Above we can see that our random forest model yielded an extremely high accuracy rate. While this may seem great, this highly suggests that we overfit our model. Let's see what some other models look like.

Decision Tree

```
set.seed(11111)
dtmod <- train(classe ~ ., method = "rpart", data = training5)
dtmodpred <- predict(dtmod, testing5)
confusionMatrix(dtmodpred, testing5$classe)$overall[1]
```

```
##  Accuracy
## 0.5258972
```

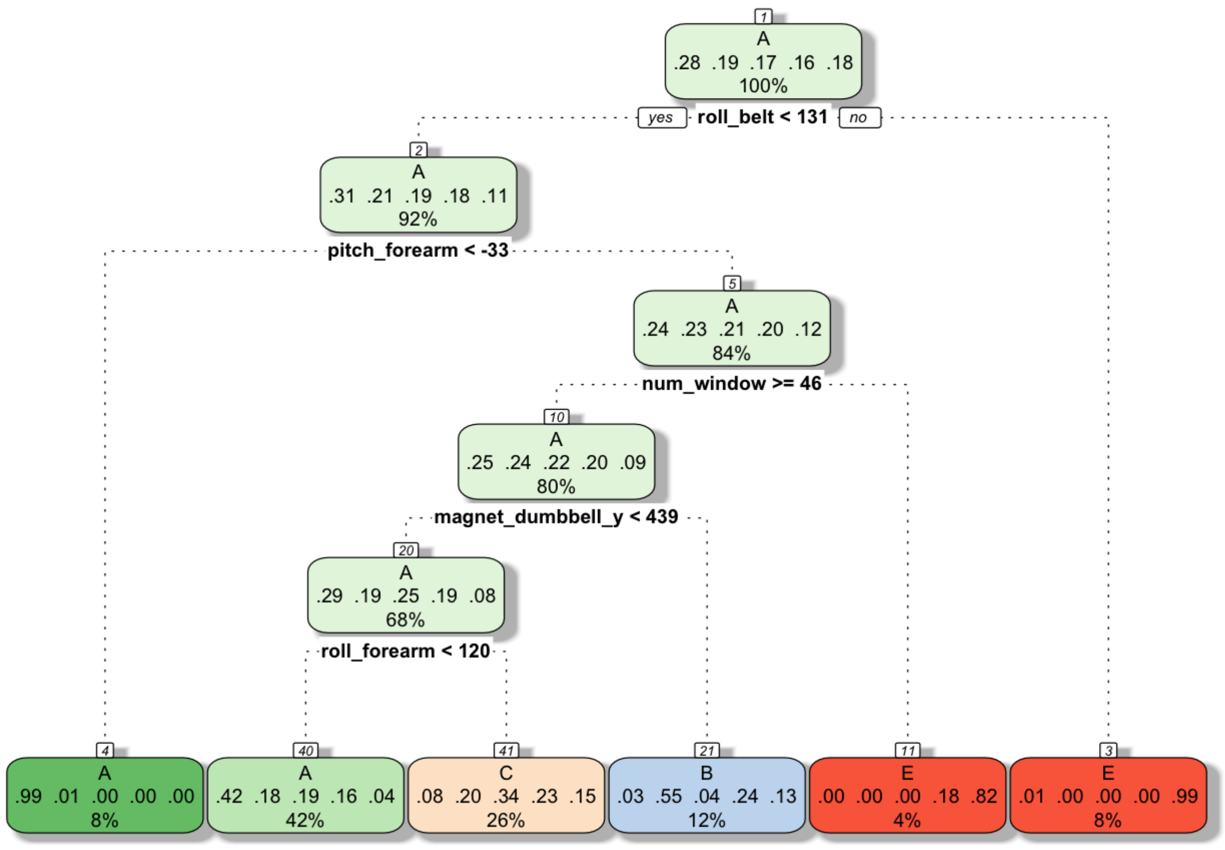
```
# get visual of deicision tree
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
fancyRpartPlot(dtmod$finalModel)
```



Rattle 2020-Feb-29 12:00:06 e.kubicek

Our decision tree model gives us an exceptionally low accuracy (~50%). Let's check out one more model before deciding which to apply to our test set.

Gradient Boosted Model

```
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
gbmod  <- train(classe ~ ., data = training5, method = "gbm",
                    trControl = control, verbose = FALSE)

gbmodpred <- predict(gbmod, testing5)
confusionMatrix(gbmodpred, testing5$classe)$overall[1]
```

```
##  Accuracy
## 0.9883768
```

Much like our random forest model, we get a 98% accuracy, this means we have a very low out of sample error. While that may seem good, it is more than likley we are overfitting. Let's see what model had the highest accuracy and fit our test data to it.

Accuracies for tested models: - Random Forest: 99.8% - Decision Tree: 49.0% - GBM: 98.6%

```
# Apply random forest model to our validation set
finaltest <- predict(rfmod, testing)
finaltest
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 
##  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
```