Practical Machine Learning Project

C. Andrés Méndez

August, 2015

Data Loading and Initial Feature Pruning

```
library(caret)
library(corrplot)
library(plyr)
```

First chunk of code where the training and testing datasets are loaded.

```
setwd("~/DataScience/MachineLearning")
traindata<-read.csv('pml-training.csv', stringsAsFactors=TRUE)
testdata<-read.csv('pml-testing.csv', stringsAsFactors=TRUE)
```

Comb and select an initial subset of apropriate features to use for classification. Avoid features with missing values and NAs.

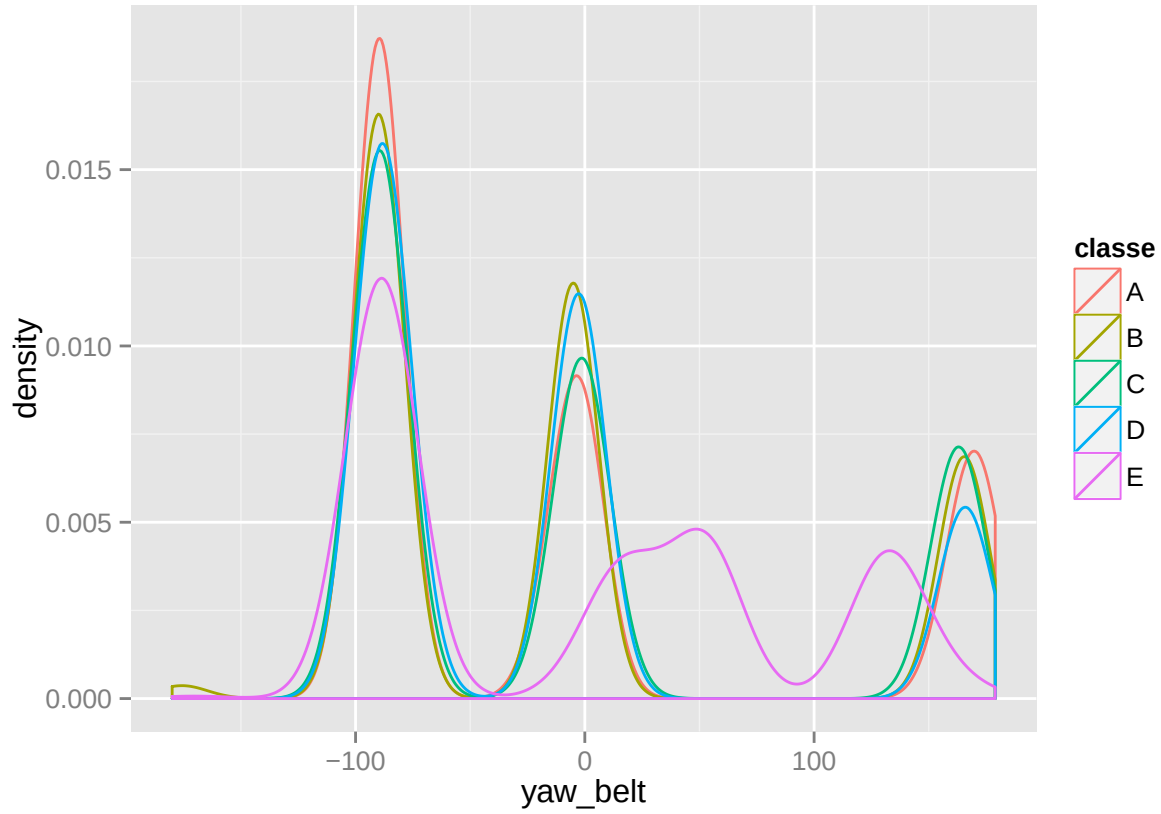
```
train_sub<-subset(traindata,select=c( "roll_belt", "pitch_belt", "yaw_belt", "total_accel_belt" ,"gyros

#create dummy variables for the factor variable "user_name" and then append the new vars to the feature
dummy_vars<-data.frame(model.matrix(~traindata$user_name))
train_sub2<-cbind(dummy_vars[,-1],train_sub)

#create the vector of target labels
train_labels=traindata$classe
```

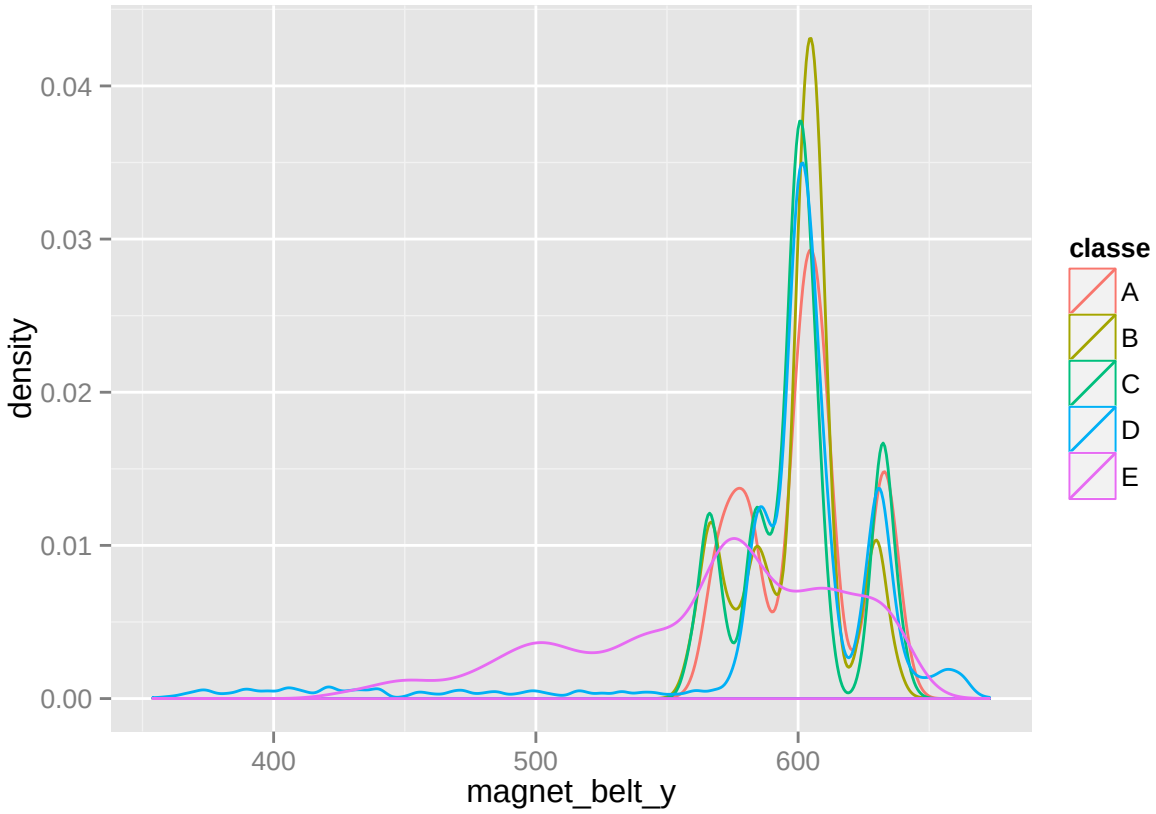
Check some features by doing scatterplots and density plots.

```
qplot(yaw_belt,colour=classe,data=train_sub2,geom="density")
```



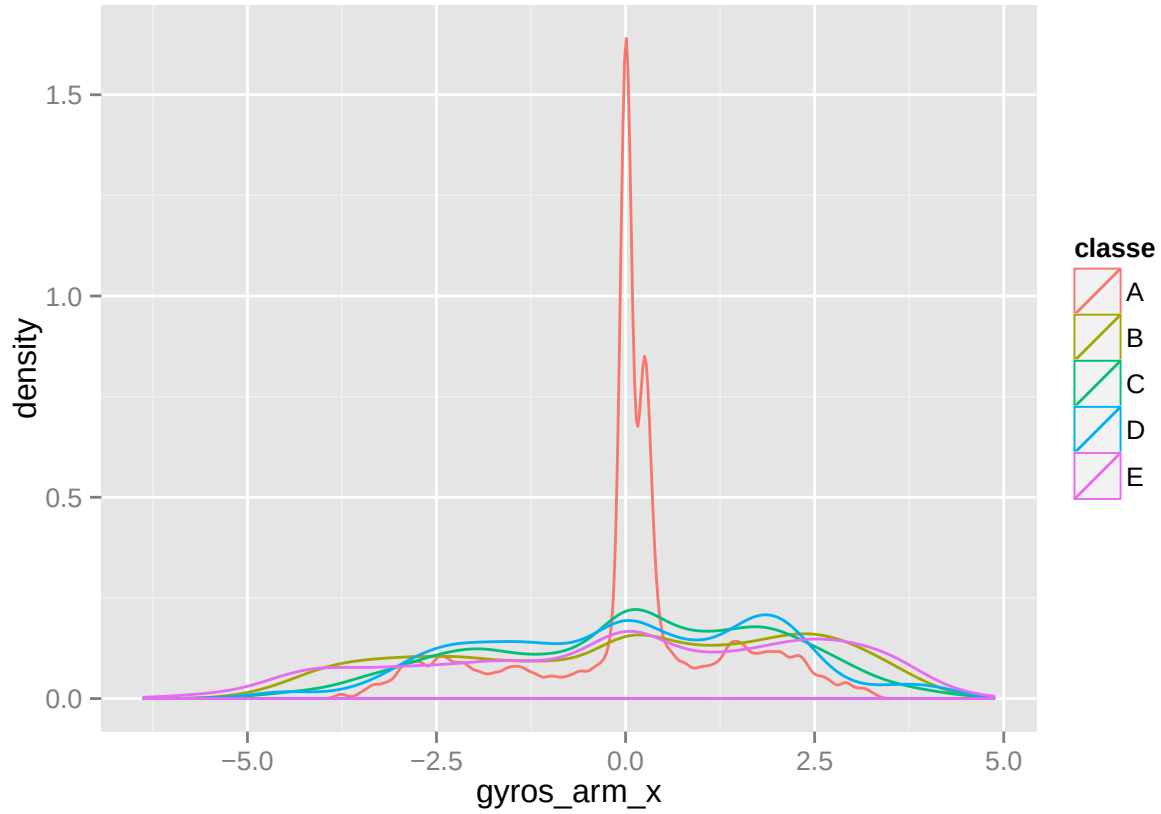
Interestingly, despite the general overlap, class E shows a wide distribution that might be useful for classification. The wide distribution of E is evident in other features such as the in the following graph.

```
qplot(magnet_belt_y,colour=classe,data=train_sub2,geom="density")
```



In some features a clear spike can be found, such as.

```
qplot(gyros_arm_x,colour=classe,data=train_sub2,geom="density")
```



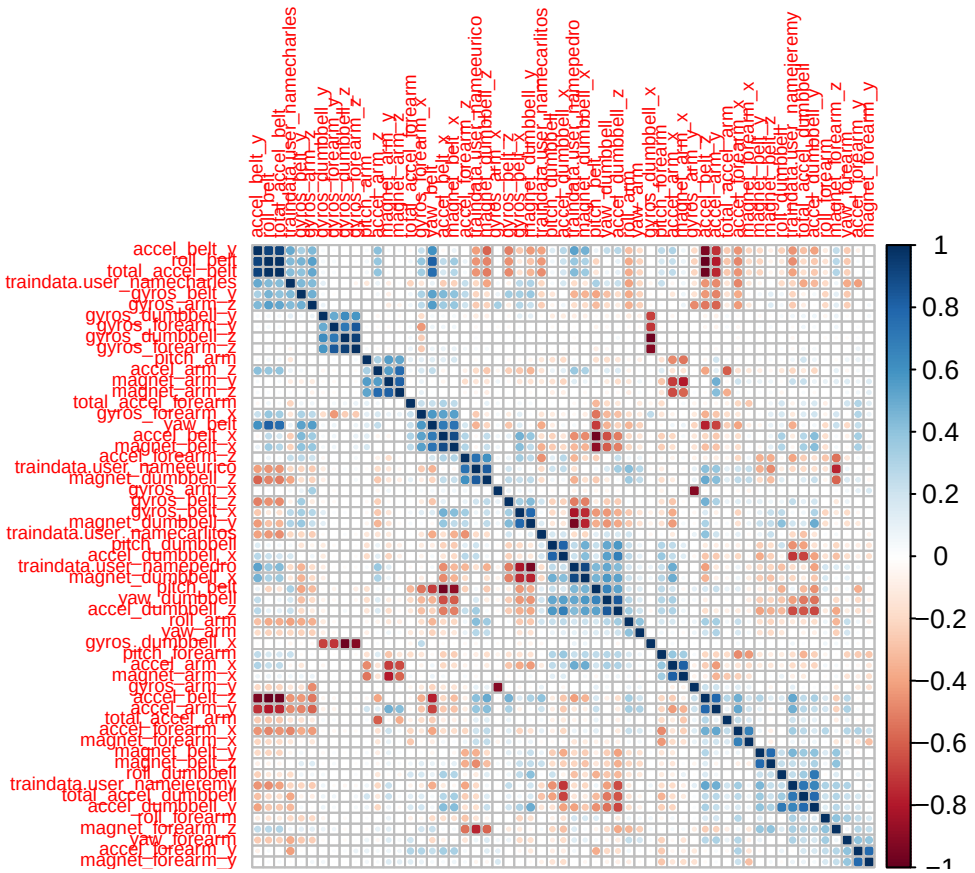
PreProcessing and Feature Selection

As a feature selection method we'll remove features with a correlation score larger than 0.7

First create a correlation matrix and visualize it ordering it via hierarchical clustering in order to visualize the clusters of correlating variables.

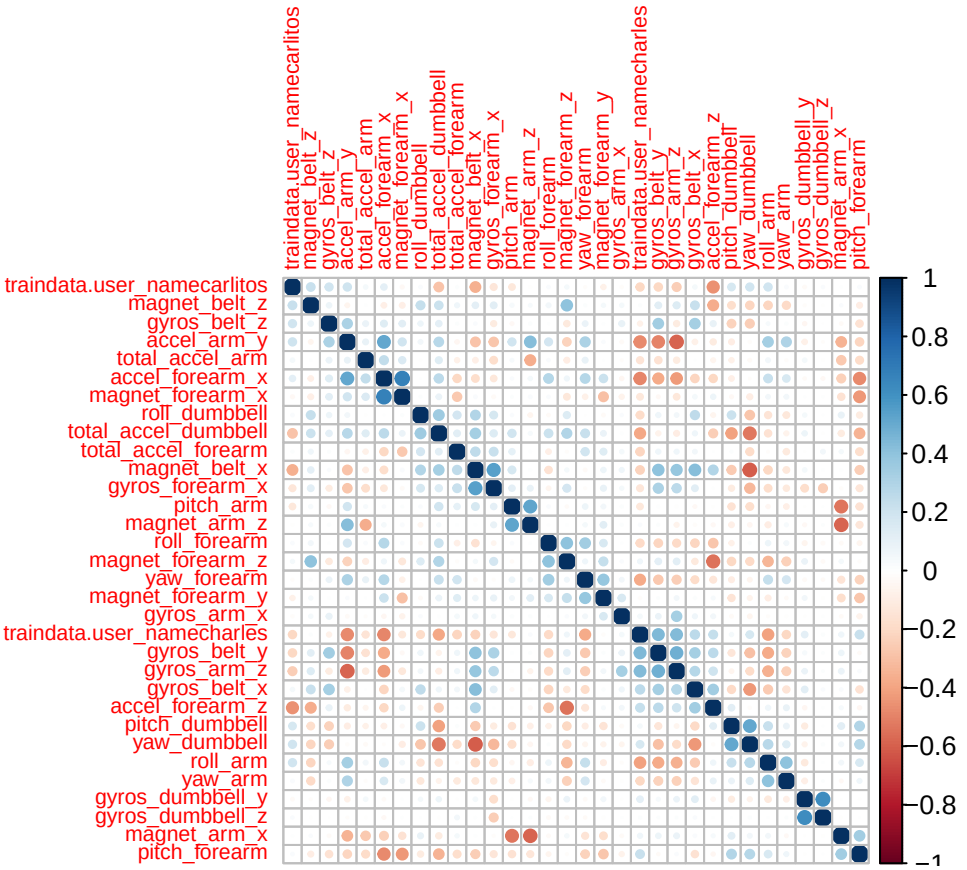
```
preproc<-preProcess(train_sub2[,-58],method=c("center","scale"))
train_preproc<-predict(preproc,train_sub2[,-58])

corMat <- cor(train_preproc)
corrplot(corMat, order = "hclust",tl.cex=.6)
```



Seeing the result we can conclude that most of the variables are not so strongly correlated. Now we set a threshold.

```
highlyCor <- findCorrelation(corMat, 0.70)
#Apply correlation filter at 0.70,
#then we remove all the variable correlated with more 0.7.
train_filtcorr<- train_preproc[,-highlyCor]
corMatfilt <- cor(train_filtcorr)
corrplot(corMatfilt, order = "hclust",tl.cex=0.65)
```




```
#add the target labels back to the preprocessed data
train_filtcorr$classe<-train_sub2$classe
```

Classification with LDA (with Bootstraping and CrossValidation)

```
#forestFit<-train(classe~.,data=train_filtcorr,method="rf",prox=TRUE)

LDAfit<-train(classe~.,data=train_filtcorr,method="lda")
```

```
## Loading required package: MASS

## Warning: package 'MASS' was built under R version 3.1.2
```

Now let's look at the characteristics of the model

```
LDAfit
```

```
## Linear Discriminant Analysis
##
## 19622 samples
##    32 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
```

```
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
##
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.5586213  0.4410136  0.005634818  0.007114673
##
##
```

The accuracy is rather low! worse than expected. Let's try using all the features (before the feture selection by correlation analysis).

```
train_preproc$classe<-train_sub2$classe
LDAfit2<-train(classe~.,data=train_preproc,method="lda")
LDAfit2
```

```
## Linear Discriminant Analysis
##
## 19622 samples
##    57 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
##
## Resampling results
##
##   Accuracy  Kappa      Accuracy SD  Kappa SD
##   0.73268   0.6614582  0.00520617   0.00657051
##
##
```

```
#Now with CROSS VALIDATION
ctrl <- trainControl(## 5-fold CV
                     method = "repeatedcv",
                     number = 5,
                     repeats=5)

LDAfit3<-train(classe~.,data=train_preproc,method="lda",trControl = ctrl)
LDAfit3
```

```
## Linear Discriminant Analysis
##
## 19622 samples
##    57 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 5 times)
##
## Summary of sample sizes: 15697, 15697, 15699, 15697, 15698, 15697, ...
##
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.7319438  0.6605381  0.007282325  0.009281652
##
##
```

The accuracy is clearly superior. For the purposes of this project let's keep this model.

Testing the model

Now apply the LDA model to the test dataset.

```
#Subset the testing data as it was done with the training data

test_sub<-subset(testdata,select=c( "roll_belt", "pitch_belt", "yaw_belt", "total_accel_belt" ,"gyros_b

dummy_vars_test<-data.frame(model.matrix(~testdata$user_name))

test_sub2<-cbind(dummy_vars_test[,-1],test_sub)
#test_sub2[,1:57] <- sapply(test_sub2[,1:57], as.numeric)

#change the name of the indicator variables to match the names in the train dataset (to avoid problems
test_sub2<-rename(test_sub2,c("testdata.user_namecarlitos"="traindata.user_namecarlitos","testdata.user

#same preproc as in training
test_preproc<-predict(preproc,test_sub2)

#with the preprocessed testing data, use the LDA model
test<-predict(LDAfit3,test_preproc)

#Show the prediction vector
test
```

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