Practical Machine Learning Proj

```
library(caret,warn.conflicts=F,verbose=F,quietly=T)
library(randomForest,warn.conflicts=F,verbose=F,quietly=T)
library(glmnet,warn.conflicts=F,verbose=F,quietly=T)
library(nnet,warn.conflicts=F,verbose=F,quietly=T)
library(gbm,warn.conflicts=F,verbose=F,quietly=T)
library(kernlab,warn.conflicts=F,verbose=F,quietly=T)
setwd("~/Downloads")
train<-read.csv("pml-training.csv",stringsAsFactors=F)
test<-read.csv("pml-testing.csv",stringsAsFactors=F)
train.feat<-train[,-c(1,2,3,4,5,6,7)]
train.feat[,-which(colnames(train.feat)=="classe")]<-suppressWarnings(sapply(train.feat[,-which(colname
train.feat$classe<-as.factor(train.feat$classe)
train.feat<-train.feat[,sapply(train.feat,function(x) all(!is.na(x)))]
index<-sample(1:nrow(train.feat),ceiling(.6*nrow(train.feat)))
train.feat.1<-train.feat[index,]
train.feat.2<-train.feat[-index,]
test.feat<-suppressWarnings(sapply(test[,colnames(test)%in%colnames(train.feat)],as.numeric))
```

The goal here is to fit a predictive classifier on a set of observations from an exercise study. The study involved repeated observations from six individuals, performing a variety of exercises. For each observation numerous meaurments were taken from accelerometers on the forearm, belt, arm, and dumbell. The outcome variable is one of 5 classes of a categorical variable that describe information on whether the exercise was done correctly.

To form a classifier, we divide the data into a training and test set. The training set consists of 19,622 observations and 160 variables (not all of which can be used in assembling a classifier) while the test set consists of 20 observations. For this prediction problem all variables were treated as numeric and features with no non missing values in the training set were removed. Furthermore, features with the majority of their observations missing were removed, resulting in only features with all observations nonmissing being included. This allowed all observations in the training set to be used instead of removing lots of observations to include all variables as some of the models used below cannot handle missing values without separate imputation.

Despite the fact that the observations are clearly hierarchical (as they are repeated observations of an individual), we ignore this in building a predictive model due to its complexity in inclusion. This obviously makes it likely that the models trained will be overfitting the data set because the individudals in the test set are different, but we proceed in this manner anyway.

The actual steps are to compare five classifiers on the training set using cross validation to pick the optimal tuning parameters. These optimal five models are then used on a validation set to decide which model is the best based on minimizing prediction error. The split between the training and validation set is 60% and 40%. We use the resulting model to predict the cases on the test set. The actual models compared are a random forest, multinomial regression, regularized multinomial regression, SVM, and a boosted model with the five previous models as inputs.

```
set.seed(26)
train_control<-trainControl(method="cv",number=10)
#random forest
rf.model<-train(classe~.,data=train.feat.1,method="rf",trControl=train_control)
#multinomial regression
multi.model<-multinom(classe~.,data=train.feat.1,trace=F)
#regularized multinomial regression
```

```
cv.lambda<-cv.glmnet(as.matrix(train.feat.1[,-ncol(train.feat.1)]),train.feat.1$classe,family="multinom
reg.model<-glmnet(as.matrix(train.feat.1[,-ncol(train.feat.1)]),train.feat.1$classe,family="multinomial
#SVM
svm.model<-train(classe~.,data=train.feat.1,method="svmLinear",trControl=train_control)
#get predictions
train.rf<-predict(rf.model)
train.multi<-predict(multi.model)
train.reg<-predict(reg.model,as.matrix(train.feat.1[,-ncol(train.feat.1)]),type="class")
train.svm<-predict(svm.model,newdata=train.feat.1[,-ncol(train.feat.1)])
#boosted model
combined.dat<-data.frame("Y"=train.feat.1$classe,"rf"=train.rf,"multi"=train.multi,"reg"=train.reg,"svm
rf.gbm<-gbm(Y~.,data=combined.dat,distribution="multinomial",cv.folds=10)
train.gbm<-predict(rf.gbm,n.trees=100,type="response")
train.gbm<-apply(train.gbm[,,],1,function(x) colnames(train.gbm[,,])[which.max(x)])
#fit on validation set
train2.rf<-predict(rf.model,newdata=train.feat.2[,-ncol(train.feat.2)])
train2.multi<-predict(multi.model,train.feat.2[,-ncol(train.feat.2)])
train2.reg<-predict(reg.model,as.matrix(train.feat.2[,-ncol(train.feat.2)]),type="class")
train2.svm<-predict(svm.model,newdata=train.feat.2[,-ncol(train.feat.2)])
combined2.dat<-data.frame("rf"=train2.rf,"multi"=train2.multi,"reg"=train2.reg,"svm"=train2.svm)
train2.gbm<-predict(rf.gbm,newdata=combined2.dat,n.trees=100,type="response")
train2.gbm<-apply(train2.gbm[,,],1,function(x) colnames(train2.gbm[,,])[which.max(x)])
```

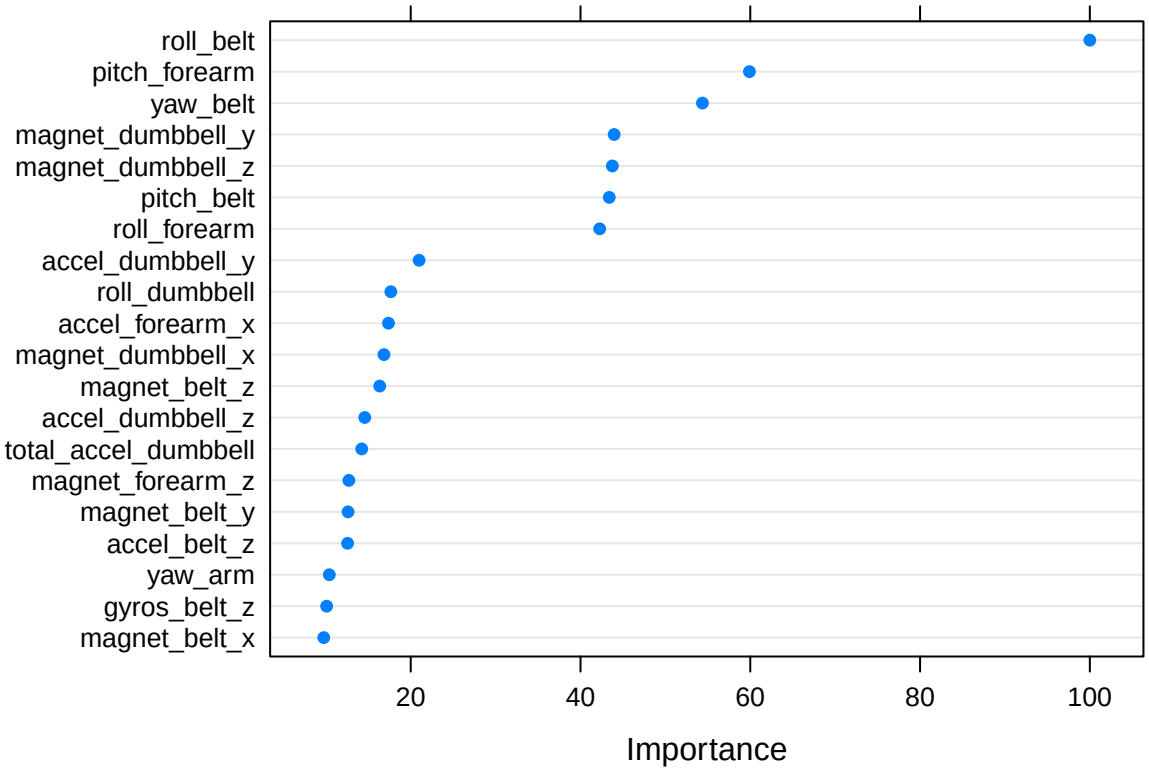
The table below shows the prediction accuracy on both the training set on which the various models were fit by cross validation, and the left out "validation" set that will be used for model selection. On the training set trees and the gbm both achieve perfect accuracy, outperforming the other methods. Note the GBM only puts weight on the random forest model output, which is why its performance is identical to the random forest model. Based on this we opt for using the random forest model since that is ultimately the method that performed the best and use it to predict the twenty test cases.

```
#computer accuracy on each data set
rf<-c(100*sum(diag(table(train.rf,train.feat.1$classe)))/sum(table(train.rf,train.feat.1$classe)),100*su
multinomial<-c(100*sum(diag(table(train.multi,train.feat.1$classe)))/sum(table(train.multi,train.feat.1
reg.multinomial<-c(100*sum(diag(table(train.reg,train.feat.1$classe)))/sum(table(train.reg,train.feat.1
svm<-c(100*sum(diag(table(train.svm,train.feat.1$classe)))/sum(table(train.svm,train.feat.1$classe)),10
gbm<-c(100*sum(diag(table(train.gbm,train.feat.1$classe)))/sum(table(train.gbm,train.feat.1$classe)),10
train.accuracy<-data.frame("rf"=rf,"multinomial"=multinomial,"reg.multinomial"=reg.multinomial,"svm"=svm
rownames(train.accuracy)<-c("train.accuracy","validation.accuracy")
train.accuracy
```

```
##                             rf multinomial reg.multinomial      svm
## train.accuracy      100.00000    65.87396        74.00204 79.17445
## validation.accuracy  98.92966    65.88940        73.72579 78.31295
##                            gbm
## train.accuracy      100.00000
## validation.accuracy  98.92966
```

The tuned random forest model used chose a random sample of 27 predictors to consider in at each split and here is a plot for the variable importance across predictors. The results of other models were omitted because they did not perform as well or were identical to the random forest (in the case of the combined boosted model).

```
dotPlot(varImp(rf.model))
```



The most important variable by far is roll_belt Pitch_forearm and yaw_belt have similar importance levels to the next four variables. Then there is a long right tail or variables with of small importance.

To approximate the out of sample error, we refit the chosen model (random forest with random sample of 27 predictors considered at each split) to the entire training data and use 10-fold cross validation as an estimate of test error.

```
rf.model.final<-train(classe~.,data=train.feat,method="rf",tuneGrid=data.frame("mtry"=27),trControl=tra
#expected test error based on CV error
100*(1-mean(rf.model.final$resample$Accuracy))
```

```
## [1] 0.4994255
```

Here are the actual predictions for the test set:

```
test.pred<-predict(rf.model.final,newdata=test.feat)
as.character(test.pred)
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```