Small Sensors - A Quantitative Study

The creation of small sensors has revolutionized the the refinement process of business. These sensors have allowed companies to gather massive amounts of data related to personnel activity. And for this specific company, they have placed sensors on six of their machinists clothes and body to determine the motions correlated with a successful product.

This study will help refine the current manufacturing process at this business, allowing the business to save time and money. Not to mention any other company that may benefit as a result of this study.

In essence, this is a binary outcome study. Either the machinists made the machine correct or incorrectly. Therefore, independent variables are compared with one another based on how they affect the dependent variable.

The dependent variable is classe, it is a factor variable that is either 1 or 0. There are a lot of independent variables, so I will not name them all here. The original data set given is train, the fully modified base dataset is dv, the model dataset is dv_var1 and the model is titled model. For the most part, every other object created is either an iteration or alteration to one those important objects.

A really big weakness is the lack of natural intuition to help guide the variable selection process. To create the ultimate dataset, every tiny little thing would have to be tested, meaning this dataset would take an enormous amount of time to fully deconstruct and build the "perfect" model. Plus, not much is given on who the company is, or who the workers are. There could be some unseen bias in the data because one machinist just happened to be significantly older than the rest, and we would never know.

The subjects in this experiment are the machinists who wore the sensors for the study. This leads back to the previous paragraph, all we really know about the subjects is that they are expert machinists, which is a pretty vague term. And there are not enough subjects to compensate for extreme abnormalities.

The independent variables in this study are the data from the accelerometers located on the machinists' belt and arm as well a bell on a manufacturing mechanism. The values are either numeric or integer though quite a few of them came up either NA or blank. The dependent variable is the binomial factor variable classe.

In the analysis I rely mainly on random forest for a few reasons. The main being the amount of time saved by using randomForest() instead of using the train funcion. It's not like randomForest() is the most accurate, especially because it has a tendency to overfit, but it will get you a good model in a short amount of time. I also use train(method='nb'), mainly because I had trouble getting train(method='rf') to work.

To start off, I needed to clean the data. The first step is simple enough, it's look at the data to see if there any abnormalities. After looking at the data, it became pretty obvious that I needed to get rid of the NA variables, which was a pretty simple process. This dramatically dropped the amount of observations in the data set. Now, looking at the data set again, I noticed that a number of the variables had missing values, though they were not considered NA values, so I missed them with my NA sweep. Getting rid of these variables proved to be somewhat of a challenge, so in the end, I used (<-NULL) on each individual variable that had a vast majority of the rows being missing. Next I removed the variables that were arbitrary.

Now that all of the variables left were useful, I needed to determine which to use.

Top variables:

	depth	vimp.all	vimp.0	vimp.1
pitch_forearm	2.709	0.057	0.042	0.096
magnet_dumbbell_z	3.169	0.066	0.044	0.121
roll_forearm	3.341	0.083	0.072	0.111
magnet_dumbbell_y	3.528	0.043	0.033	0.069
roll_arm	3.934	0.035	0.034	0.039
magnet_arm_x	4.088	0.031	0.017	0.065
total_accel_dumbbell	4.099	0.028	0.018	0.052
magnet_dumbbell_x	4.162	0.037	0.022	0.075
accel_arm_x	4.331	0.028	0.018	0.053
pitch_arm	4.542	0.026	0.024	0.029
accel_dumbbell_z	4.542	0.021	0.016	0.032
accel_forearm_x	4.694	0.019	0.011	0.039
magnet_forearm_x	4.752	0.018	0.008	0.042
pitch_belt	4.800	0.029	0.012	0.069
accel_dumbbell_y	4.980	0.017	0.012	0.029
yaw_dumbbell	5.020	0.020	0.013	0.036
total_accel_forearm	5.062	0.011	0.006	0.025
magnet_arm_z	5.107	0.012	0.006	0.027
magnet_arm_y	5.141	0.018	0.013	0.030
magnet_forearm_z	5.156	0.014	0.010	0.024
accel_forearm_z	5.171	0.017	0.013	0.026
roll_belt	5.187	0.032	0.027	0.045
yaw_arm	5.247	0.014	0.011	0.022
roll_dumbbell	5.380	0.013	0.010	0.020
magnet_forearm_y	5.464	0.011	0.008	0.019
accel_dumbbell_x	5.555	0.012	0.008	0.022
accel_belt_z	5.589	0.014	0.008	0.028
yaw_forearm	5.630	0.012	0.009	0.019
accel_arm_y	5.636	0.013	0.009	0.023
magnet_belt_z	5.758	0.010	0.005	0.023
gyros_dumbbell_y	5.849	0.007	0.003	0.016
magnet_belt_y	5.886	0.009	0.004	0.020
magnet_belt_x	5.916	0.012	0.008	0.022
gyros_arm_x	5.944	0.009	0.003	0.023
accel_arm_z	5.968	0.009	0.005	0.019
accel_forearm_y	5.996	0.008	0.005	0.017
total_accel_belt	6.206	0.012	0.007	0.024
gyros_arm_y	6.270	0.006	0.002	0.014
total_accel_arm	6.299	0.007	0.003	0.015
accel_belt_x	6.385	0.008	0.004	0.018

pitch_dumbbell	6.403	0.006	0.005	0.011
gyros_forearm_y	6.448	0.005	0.002	0.012
gyros_dumbbell_x	6.784	0.003	0.002	0.006
gyros_belt_x	6.828	0.005	0.002	0.011
gyros_forearm_x	6.869	0.003	0.002	0.005
gyros_forearm_z	6.907	0.002	0.001	0.005
gyros_arm_z	6.947	0.003	0.002	0.007
gyros_belt_z	7.007	0.006	0.002	0.015
accel_belt_y	7.107	0.010	0.005	0.021
gyros_dumbbell_z	7.163	0.002	0.002	0.004
gyros_belt_y	7.847	0.006	0.003	0.013

This is the table I used to determine what variables would go in my model. The problem with having so many variables is you lose the ability and time to check each variable. The higher up, the more impact the variable has.

The original sample 160 variables long and had 19622 observations. And through a long process, those numbers went down 52 variables and 13816 observations.

I would say my analysis was a success. I found a model with a success rate of 0.82 for finding if the the machine was not built properly, and a 0.89 for predicting if the machine was built properly.

Based on these results, it is fairly conclusive that the way a machinists works affects productivity, which could potentially save the company a lot of money.

Majority of Project Code

```
#open csv files
test <- read.csv('machine-testing.csv')
train <- read.csv('machine-training.csv')

# Explore the dataset and determine which of the predictors are worth keeping in the analysis.
# Eliminate NA columns and create data set to test
ds <- train[ , colSums(is.na(train)) < 9811]
# Convert classe into dichotomous outcome
ds$classe <- ifelse(ds$classe=='A', 1, 0)

#long method
dv <- ds
summary(dv)
  #individually remove all empty variables
dv$kurtosis_yaw_belt<-NULL
dv$kurtosis_roll_belt<-NULL
dv$skewness_roll_belt<-NULL
dv$skewness_roll_belt<-NULL
dv$yaw_belt<-NULL
dv$max_yaw_belt<-NULL
dv$min_yaw_belt<-NULL
dv$amplitude_yaw_belt<-NULL
dv$kurtosis_roll_arm<-NULL
dv$kurtosis_picth_arm<-NULL
dv$kurtosis_yaw_arm<-NULL
dv$skewness_roll_arm<-NULL
dv$skewness_pitch_arm<-NULL
dv$skewness_yaw_arm<-NULL
dv$kurtosis_roll_dumbbell<-NULL
dv$kurtosis_picth_dumbbell<-NULL
dv$kurtosis_yaw_dumbbell<-NULL
dv$skewness_roll_dumbbell<-NULL
dv$skewness_pitch_dumbbell<-NULL
dv$skewness_yaw_dumbbell<-NULL
dv$max_yaw_dumbbell<-NULL
dv$min_yaw_dumbbell<-NULL
dv$amplitude_yaw_dumbbell<-NULL
dv$kurtosis_roll_forearm<-NULL
dv$kurtosis_picth_forearm<-NULL
dv$kurtosis_yaw_forearm<-NULL
dv$skewness_roll_forearm<-NULL
dv$skewness_pitch_forearm<-NULL
dv$skewness_yaw_forearm<-NULL
dv$max_yaw_forearm<-NULL
dv$min_yaw_forearm<-NULL
dv$amplitude_yaw_forearm<-NULL
 #missed variables
```

```
dv$kurtosis_picth_belt<-NULL
dv$skewness_roll_belt.1<-NULL
dv$skewness_yaw_belt<-NULL

#determine if non-variable/integer variables should stay
theory = dv
table(theory$classe[theory$user_name=="adelmo"])
table(theory$classe[theory$user_name=="carlitos"])
table(theory$classe[theory$user_name=="charles"])
table(theory$classe[theory$user_name=="eurico"])
table(theory$classe[theory$user_name=="jeremy"])
table(theory$classe[theory$user_name=="pedro"])
theory$raw_timestamp_part_1 <-NULL
theory$user_name<-NULL
theory$raw_timestamp_part_2<-NULL
theory$cvtd_timestamp<-NULL
theory$new_window<-NULL
theory$num_window<-NULL
theory$x<-NULL

dv = theory
# Data Partition
set.seed(425)
ind_dv <- sample(2, nrow(dv), replace = TRUE, prob = c(0.7, 0.3))
train_dv <- dv[ind_dv==1,]
test_dv <- dv[ind_dv==2,]
train_dv <- as.data.frame(train_dv)
train_dv$classe <- as.factor(train_dv$classe)
#train_2 <- train_dv.combine[1:13816, c("pitch_forearm", "magnet_dumbbell_z",
            "roll_forearm", "magnet_dumbbell_y", "roll_arm magnet_arm_x")]

#Variable Selection
var.select(classe~., train_dv)
#dv_var1 <- data.frame(dv$pitch_forearm,dv$magnet_dumbbell_z,dv$roll_forearm,
#                       dv$magnet_dumbbell_y,dv$roll_arm,dv$magnet_arm_x,
 #                      dv$classe)
dv_var1 <- train_dv[c("pitch_forearm","magnet_dumbbell_z","roll_forearm",

"magnet_dumbbell_y","roll_arm","magnet_arm_x","total_accel_dumbbell",
                    "classe")]
dv_var2 <- data.frame(dv$gyros_belt_z,dv$gyros_forearm_x,dv$gyros_belt_z,dv$classe)
dv_var1$classe <- as.factor(dv_var1$classe)
# Random Forest
library(randomForest)
set.seed(573)
rf <- randomForest(classe~., data = train_dv)
print(rf)
dim(rf)
```

```
  #Testing Variables
set.seed(835)
rf_var1 <- randomForest(classe~., data=dv_var1)
rf_var2 <- randomForest(dv.classe~., data=dv_var2)


# Prediction & Confusion Matrix - train data
library(caret)
p1 <- predict(rf, train_dv)
confusionMatrix(p1, train_dv$classe)
 #Variable Selection
  p_var1 <- predict(rf_var1, dv_var1)
  p_var2 <- predict(rf_var2, dv_var2)
  confusionMatrix(p_var1, dv_var1$classe)
  confusionMatrix(p_var2, dv_var2$dv.classe)#Var1 ended up way better
# Prediction & Confusion Matrix - test data
p2 <- predict(rf, test_dv)
confusionMatrix(p2, test_dv$classe)
p2_var1 <- predict(rf_var1, test_dv)
confusionMatrix(p2_var1, test_dv$classe )
# Error rate of Random Forest
plot(rf)


# Cross-Validation
set.seed(6872)
cv.10.folds <- createMultiFolds(dv$classe, k=10, times = 10)


# Set up caret's trainControl object
ctrl.1 <- trainControl(method = "repeatedcv", number = 10, repeats = 10, index =
cv.10.folds)


#Start-up doSnow
library(doSNOW)
cl <- makeCluster(8, type = "SOCK")
registerDoSNOW(cl)
  #shutdown doSnow after train
stopCluster(cl)


# Set seed for reproducibility and train
set.seed(4762)
rf.cv.1 <- train(x = dv, y=dv$classe, method="rf", tuneLenght = 3,
                 ntree = 1000, trControl = ctrl.1)
```

Majority of Project Code

```
#Build Model
model <- rpart(classe ~ pitch_forearm+magnet_dumbbell_z+roll_forearm+
               magnet_dumbbell_y+roll_arm+magnet_arm_x+
               total_accel_dumbbell, method = "class", data = dv_var1)

#Test for Out of Sample Error
set.seed(7358)
tr_control <- trainControl(method = "boot", number = 100)
boot_strap <- train(classe~., data = dv_var1, trControl=tr_control, method='nb')
print(boot_strap)

# Test the test data
ptest <- predict(rf,test)
confusionMatrix(ptest, test, type="response")

prediction <- predict(model,test)

var1_test <- predict(rf_va)
```