# DrivenData Challenge: Predicting Blood Donations

**Joanne Breitfelder**

***20 Sep 2016***

---

## Introduction

*Blood donation has been around for a long time. The first successful recorded transfusion was between two dogs in 1665, and the first medical use of human blood in a transfusion occurred in 1818. Even today, donated blood remains a critical resource during emergencies. The dataset is from a mobile blood donation vehicle in Taiwan. The Blood Transfusion Service Center drives to different universities and collects blood as part of a blood drive. We want to predict whether or not a donor will give blood the next time the vehicle comes to campus.*

This challenge is proposed by DrivenData.

Data is courtesy of Yeh, I-Cheng via the UCI Machine Learning repository:
Yeh, I-Cheng, Yang, King-Jang, and Ting, Tao-Ming, *Knowledge discovery on RFM model using Bernoulli sequence*, **Expert Systems with Applications**, 2008, doi:10.1016/j.eswa.2008.07.018.

**Ranking (team DataXplorers):**
- Feb. 27, 2016: #1
- Sept. 20, 2016: #5

---

## Data exploration

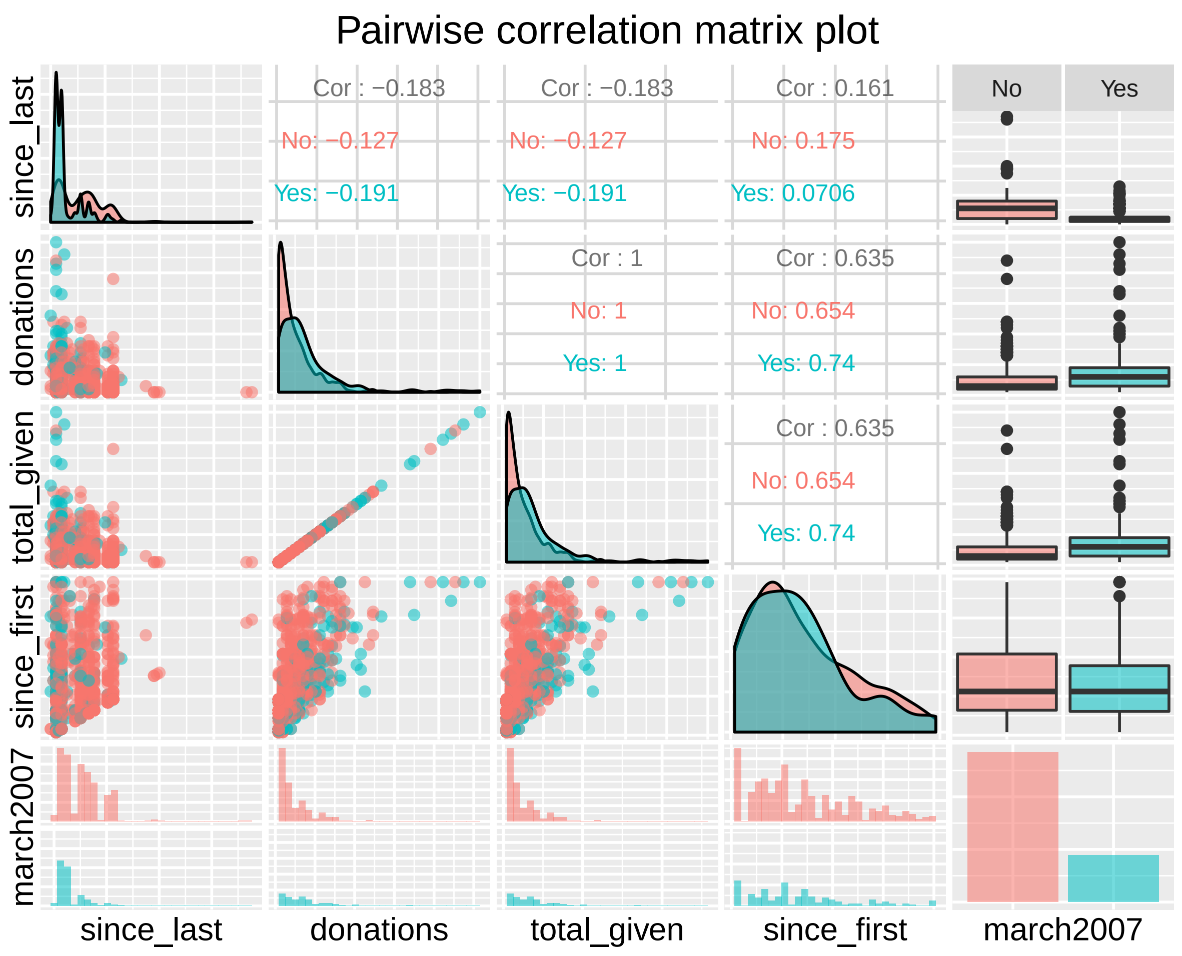
Let's first have a look at the `train` dataset:

| since_last | donations | total_given | since_first | march2007 |
|---:|---:|---:|---:|:---|
| 2 | 50 | 12500 | 98 | Yes |
| 0 | 13 | 3250 | 28 | Yes |
| 1 | 16 | 4000 | 35 | Yes |
| 2 | 20 | 5000 | 45 | Yes |
| 1 | 24 | 6000 | 77 | No |

The different variables are:

- **since_last**: number of months since the last donation,
- **donations**: total number of donations,
- **total_given**: total blood donated in c.c.,
- **since_first**: number of months since the first donation,
- **march2007**: a binary variable representing whether he/she donated blood in March 2007.

The goal of the challenge is to predict the last column, whether he/she donated blood in March 2007. Let's begin by looking at the correlations between the different variables.



From this graphic we can conclude at least 2 important things:

- **total_given** and **donations** are highly correlated,
- some variables have highly skewed distributions.

---

## Features ingeneering

Let's introduce some new variables:

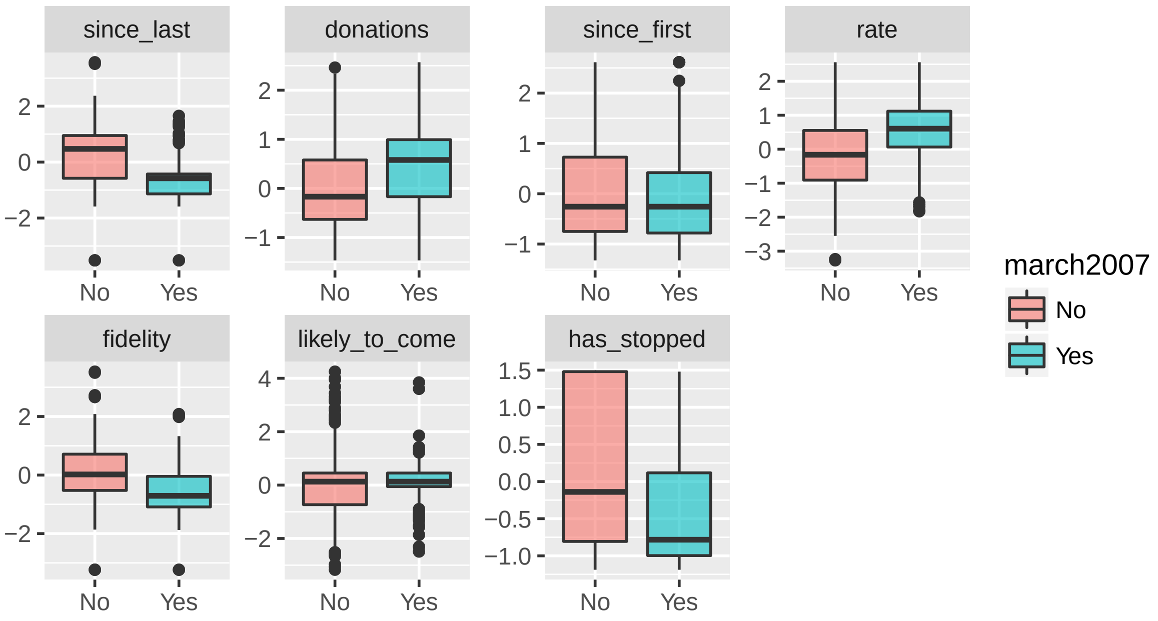
- **rate**: donations frequence.
- **fidelity**: if this number is small, it indicates that the subject has made a lot of donations, including recent ones.
- **likely_to_come**: if he/she comes every 3 months in average and came for the last time 3 months ago, we can assume that he/she is likely to come in March 2007, and then this variable is close to 0.
- **has_stopped**: a value close to 1 indicates a person who has probably given up donating blood.

```
train <- mutate(train, rate=donations/since_first,
                fidelity=since_last/donations,
                likely_to_come=1/rate-since_last,
                has_stopped=since_last/since_first)
```

Let's remove the total quantity (which is highly correlated with the number of donations), apply a boxcox transformation on the skewed variables and normalize the data:

```r
train <- train[, -3]
train <- train[, c(1, 2, 3, 5, 6, 7, 8, 4)]

my_norm <- function(x) {x <- (x-mean(x))/sd(x)}

my_boxcox <- function(x) {
        if(sum(x==0)!=0) {x <- x+1e-6} ## Adding a small value to avoid zeros
        bc <- BoxCoxTrans(x)
        L <- bc$lambda
        if(L!=0) {x <- (x^L-1)/L}
        if(L==0) {x <- log(x)}
        return(x)
}

train[, c(1, 2, 4, 5)] <- lapply(train[, c(1, 2, 4, 5)], my_boxcox)
train[, -8] <- lapply(train[, -8], my_norm)
```

Let's have a look at the resulting variables:



Some variables seem to be better at explaining the output, but we will include all of them in the model anyway. Of course we apply exactly the same transformations on the test dataset.

---

## Machine Learning

The whole Machine Learning part is treated with the `caret` package.

### Splitting the data

In order to test the results of our Machine learning model, it is a good idea to create a validation dataset.

```
index <- createDataPartition(y=train$march2007, p=0.7, list=FALSE)
train <- train[index, ]
validation <- train[-index, ]
```

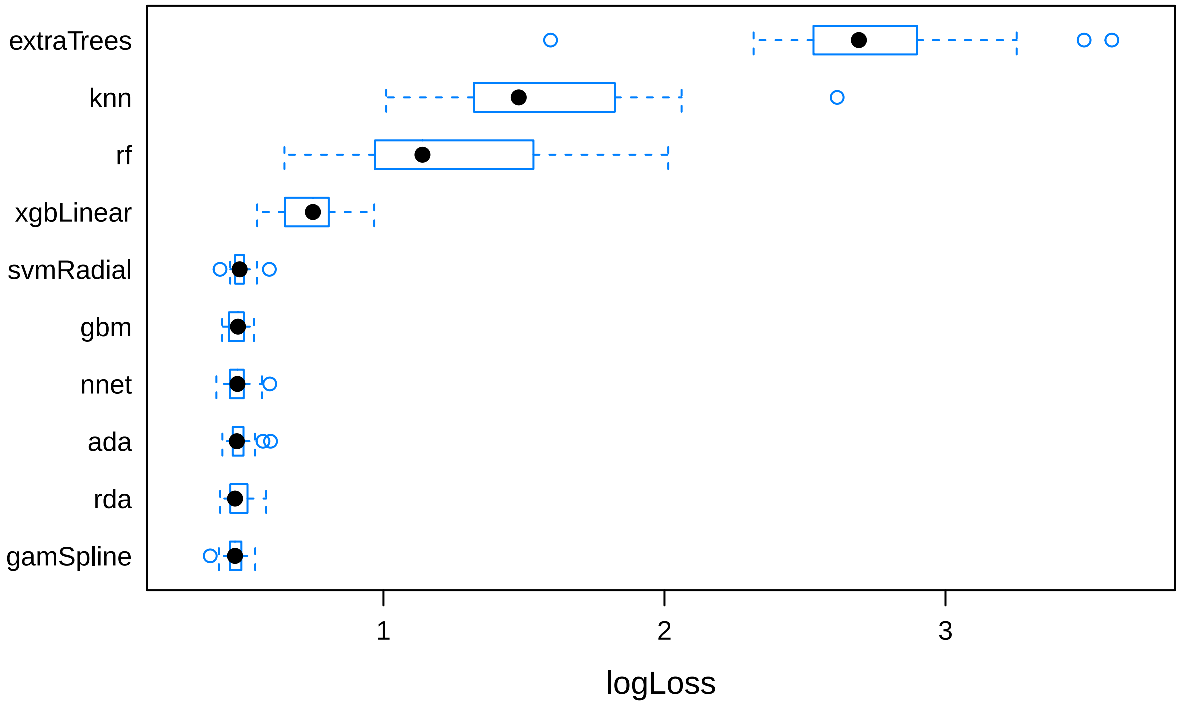
### Training strategy

The `trainControl` function allows to control the computational nuances of the `train` function. Here we first set the resampling method to avoid over-fitting (K-fold cross-validation with K=10, repeated 10 times). Then we precise that we will be **predicting a probability**, and we will use the **Log Loss** to evaluate the performance metrics.

```
tc <- trainControl(method="repeatedcv", number=10, repeats=10,
                   classProbs=TRUE, summaryFunction=mnLogLoss,
                   allowParallel=TRUE)
```

### Machine learning models

I first test a bunch of different models to select the best ones.



Let's keep the following models: **gbm** (Generalized boosted regression), **ada** (Stochastic boosting Model), **gamSpline** (Generalized additive model with integrated smoothness estimation), **svmRadial** (Support vector machine), et **nnet** (Single-hidden-layer neural network). To improve the performance, we can search for the best parameters through tuning grids.

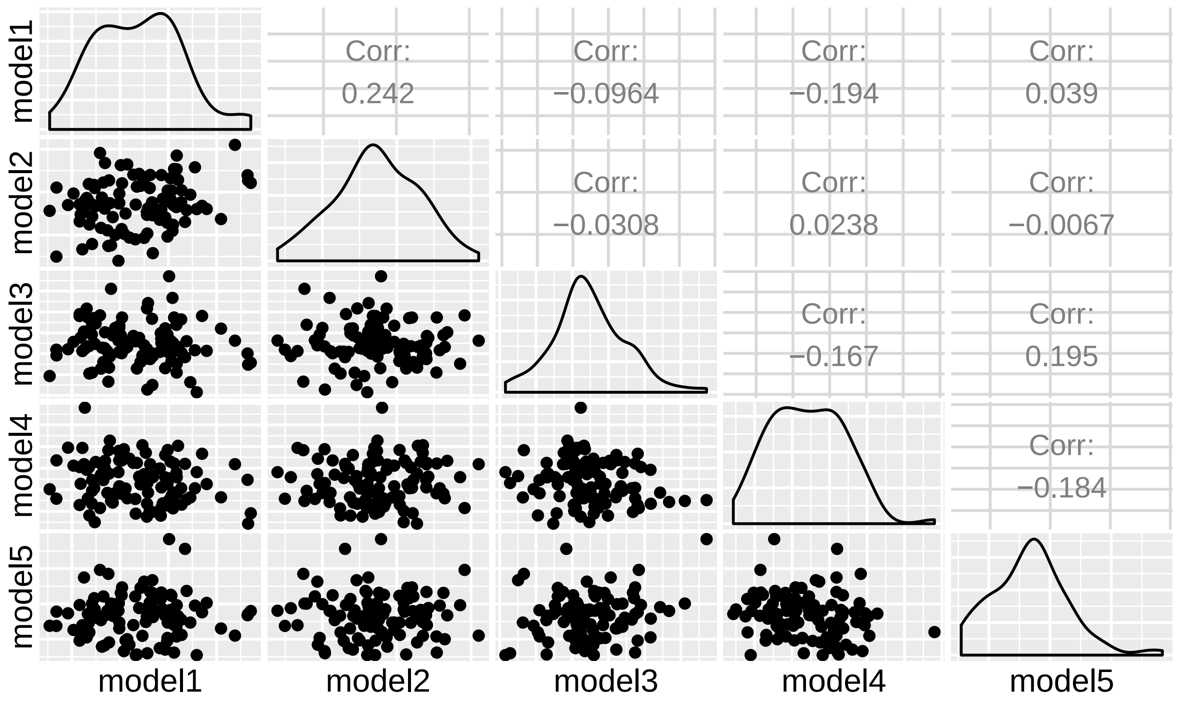
```
gbmGrid <- expand.grid(n.trees=500,
                       interaction.depth=2,
                       shrinkage=0.005,
                       n.minobsinnode=10)

nnetGrid <- expand.grid(.decay=c(0.1, 0.5), .size=c(3, 4, 5))
```

```
svmGrid <- expand.grid(sigma=c(0.5),
                       C=c(0.3))

adaGrid <- expand.grid(iter=100, maxdepth=3,
                       nu=0.05)
```

We can now train the models. Pre-processing the data by applying a **principal component analysis** slightly improves the final results.

```
model1 <- train(march2007 ~., train, preProcess=c("pca"), method='gbm',
                metric="logLoss", maximize=FALSE, trControl=tc,
                verbose=FALSE, tuneGrid=gbmGrid)

model2 <- train(march2007 ~., train, preProcess=c("pca"), method='ada',
                metric="logLoss", maximize=FALSE, tuneGrid=adaGrid,
                trControl=tc)

model3 <- train(march2007 ~., train, preProcess=c("pca"), method='svmRadial',
                metric="logLoss", maximize=FALSE, trControl=tc,
                tuneGrid=svmGrid)

model4 <- train(march2007 ~., train, preProcess=c("pca"), method="nnet",
                tuneGrid=nnetGrid, maxit=1000,
                trControl=tc, metric="logLoss", trace=FALSE)

model5 <- train(march2007 ~., train, preProcess=c("pca"), method="gamSpline",
                trControl=tc, metric="logLoss")
```

Let's check the correlations between the different models:



All the models are rather weakly correlated, what make them good candidates for being combined in an ensemble prediction. Indeed, it means that they all explain different aspects of the data.

### Ensemble modeling

The strategy is now to combine the results given by the different models and train a new model over them. Let's make the 5 predictions and use them to build a new dataframe.

```r
pred1V <- predict(model1, train, "prob")
pred2V <- predict(model2, train, "prob")
pred3V <- predict(model3, train, "prob")
pred4V <- predict(model4, train, "prob")
pred5V <- predict(model5, train, "prob")

combined.data <- data.frame(pred1=pred1V, pred2=pred2V, pred3=pred3V,
                            pred4=pred4V, pred5=pred5V,
                            march2007=train$march2007)
```

We are now ready to train a new algorithm, in this case I choose a general boosted regression model.

```r
gbmGrid <- expand.grid(n.trees=500, interaction.depth=3,
                       shrinkage=0.01, n.minobsinnode=10)

combined.model <- train(march2007 ~., combined.data, method='gbm',
                  metric="logLoss", maximize=FALSE, trControl=tc,
                  verbose=FALSE)

combined.result <- predict(combined.model, combined.data, "prob")
combined.result$obs <- train$march2007
logloss = mnLogLoss(combined.result, lev=levels(combined.result$obs))
```

On the training dataset we get a rather good Log Loss of 0.3301318.

### Validation

We now have to check if the results are also good on the validation dataset. If not, it will probably mean that we are over-fitting.

```r
pred1V <- predict(model1, validation, "prob")
pred2V <- predict(model2, validation, "prob")
pred3V <- predict(model3, validation, "prob")
pred4V <- predict(model4, validation, "prob")
pred5V <- predict(model5, validation, "prob")

combined.data <- data.frame(pred1=pred1V, pred2=pred2V, pred3=pred3V,
                            pred4=pred4V, pred5=pred5V,
                            march2007=validation$march2007)

combined.result <- predict(combined.model, combined.data, "prob")
combined.result$obs <- validation$march2007
logloss <- mnLogLoss(combined.result, lev=levels(combined.result$obs))
```

Applied on the validation dataset, our model gives a Log Loss of 0.3527234. The log loss is slightly less good on the validation dataset, but still in the same range.

### Prediction on test cases

We are now ready to get results on the test cases given on the DrivenData website:

| | Made Donation in March 2007 |
|---|---:|
| 659 | 0.5065938 |
| 276 | 0.0462945 |
| 263 | 0.2900445 |
| 303 | 0.1631766 |
| 83 | 0.7353806 |