Predicting Default of Credit Card Holders Using Additional Data

Valerii Podymov

June 18, 2017

Overview

This time we have two additional data sets. One of them describes 55 agents and the other describes 34103 calls that they have placed. Some debtors were called by multiple agents before they repaid or their account was abandoned.

The original data set comes from UCI Machine Learning Repository and also were used in the Kaggle's Default of Credit Card Clients challenge.

Our goal is to use additional knowledge to improve the model and still keep it as compact as it possible.

Data Cleaning and Feature Engineering

First of all we need to provide the right variable types (numeric, date-time of factor) and remove some exra variables.

```r
#### Load data
raw_df <- read.csv('UCI_Credit_Card.csv')
ag_df <- read.csv('agents.csv')
call_df <- read.csv('calls.csv')

sum(is.na(ag_df)); sum(is.na(call_df)) ## check missing values
```

```
## [1] 0
```

```
## [1] 0
```

The new data sets do not contain missing values.

Then we assign meaningful names to variables and provide necessary type conversions.

```r
names(raw_df) <- tolower(names(raw_df)) # convert to lowercase
names(raw_df)[names(raw_df) =="id"] <- "debtor_id"
names(raw_df)[names(raw_df) =="default.payment.next.month"] <- "output"
raw_df$output <- as.factor(raw_df$output)
raw_df$sex <- as.factor(raw_df$sex)
raw_df$marriage <- as.factor(raw_df$marriage)
raw_df$debtor_id <- as.factor(raw_df$debtor_id)

extra6 <- raw_df$education == 6
raw_df[extra6, 'education'] <- 5
raw_df$education <- as.factor(raw_df$education)

names(ag_df)[names(ag_df) =="id"] <- "agent_id"
ag_df$agent_id <- as.factor(ag_df$agent_id)
ag_df$call_centre_id <- as.factor(ag_df$call_centre_id)
ag_df$supervisor_id <- as.factor(ag_df$supervisor_id)
```

```r
ag_df$hire_date <- as.Date(ag_df$hire_date, format = "%m-%d-%y") # to date

call_df$agent_id <- as.factor(call_df$agent_id)
```

Let's take a closer look at `call_df` data set. We now, that some debtors were called multiple times. To find how many times each debtor was called, we do the following.

```r
a <- rle(sort(call_df$debtor_id)) # run length encoding
cc_count <- data.frame(debtor_id = a$values, call_count = a$lengths)

call_df$debtor_id <- as.factor(call_df$debtor_id)
cc_count$debtor_id <- as.factor(cc_count$debtor_id)
```

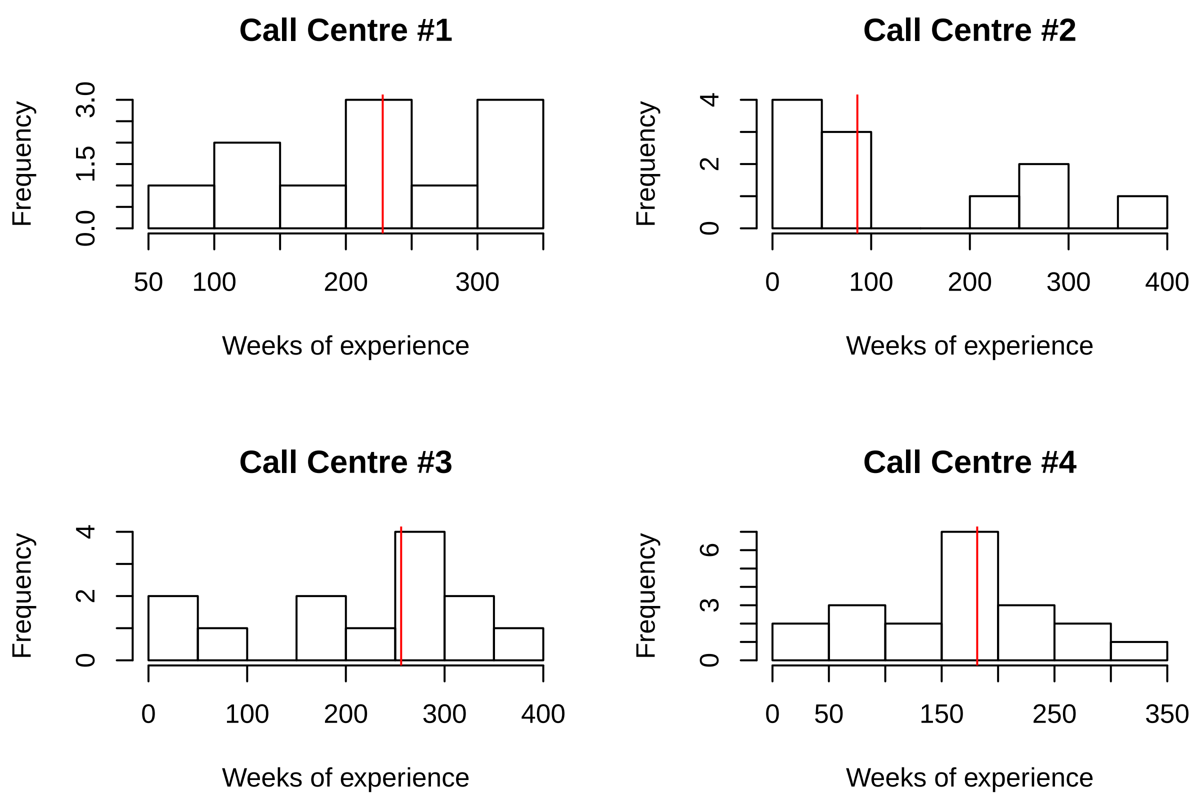
The `cc_count` data frame contains 20510 observations. That means not all the debtors in the initial data set `raw_df` were called.

We do not have information on time or sequences the calls were placed and therefore cannot say when agents placed their call and which of the calls were successful.

However in the `ag_df` dats set there is information about the hire date, so we can get an idea about the experience and assume that qualification of a certain agent may influence the outcome.

We can describe the agent experience by new variable which shows the whole number of weeks passed since the hire date.

```r
today <- Sys.Date()  # today's date
ag_df$experience <- floor(difftime(today, ag_df$hire_date,
                              units = "weeks"))
ag_df$experience <- as.numeric(ag_df$experience)

#### compare histograms of agents experience
cc_1 <- filter(ag_df, call_centre_id == '1')
cc_2 <- filter(ag_df, call_centre_id == '2')
cc_3 <- filter(ag_df, call_centre_id == '3')
cc_4 <- filter(ag_df, call_centre_id == '4')

par(mfrow = c(2, 2))

hist(cc_1$experience, xlab = "Weeks of experience",
     main = "Call Centre #1")
abline(v = median(cc_1$experience), col = "red")

hist(cc_2$experience, xlab = "Weeks of experience",
     main = "Call Centre #2")
abline(v = median(cc_2$experience), col = "red")

hist(cc_3$experience, xlab = "Weeks of experience",
     main = "Call Centre #3")
abline(v = median(cc_3$experience), col = "red")

hist(cc_4$experience, xlab = "Weeks of experience",
     main = "Call Centre #4")
abline(v = median(cc_4$experience), col = "red")
```



As it shown on the histograms above, all the call centres are different in terms of the median values (red vertical lines) of the personnel experience measured in the number of weeks from the hire date.

We can demonstrate that the values of `agent_id`, `supervisor_id`, `call_centre_id` variables in the `ag_df` data set reflect the certain hirachy.

For example, for `call_centre_id` equal to `1` we can see only two corresponding values of `supervisor_id`, `3` and `4`.

```r
summary(cc_1$supervisor_id)
```

```
##  3  4  5  6  7  8  9 10
##  5  6  0  0  0  0  0  0
```

Similarly for `call_centre_id` equal to `2` the corresponding values of `supervisor_id` are `5` and `6` etc.

```r
summary(cc_2$supervisor_id)
```

```
##  3  4  5  6  7  8  9 10
##  0  0  8  3  0  0  0  0
```

In the same way we can demonstrate that certain `agent_id`s correspond to certain `supervisor_id`s. It means that by `agent_id` we can clearly identify both `supervisor_id` and `call_centre_id`.

Assume that the positive outcome generated by certain agents depends on many factors such as their experience, the maturity of processes in the call centres etc. By this reason we can use `call_centre_id` factor variable as a consolidated set of explicit and hidden factors influencing the performance of a certain call centre and the ability of its employees to generate a positive outcome. By this way we will avoid a redundancy of the further model caused by huge number of levels in factor variables. Thus, `name`, `hire_date`, `supervisor_id` and `experience` columns would be extra.

```
#### Get rid off extra variables
ag_df$name <- NULL
ag_df$hire_date <- NULL
ag_df$supervisor_id <- NULL
ag_df$experience <- NULL
```

Now we need to merge different data sets into one data frame. We start from merging `call_df` and `ag_df` data frames to find which call centres contacted certain debtors.

```
#### join call_df and ag_df
deb_cc <- merge(call_df, ag_df, by = 'agent_id')
deb_cc$agent_id <- NULL
```

It can be easily noted that some debtors were contacted by different call centres. Because of this there are duplicated `debtor_id`s in the `deb_cc`, so this data frame is not clean. To make it cleaner we will dummify `call_centre_id` variable using one-hot encoding and then group data by `debtor_id`.

```
deb_cc_dummied <- dummy.data.frame(deb_cc,
                                  names = "call_centre_id", sep="")
deb_cc_agg <- aggregate(.~ debtor_id, deb_cc_dummied, sum)
```

Data frame `deb_cc_agg` has 20510 observations, same as `cc_count` data frame. The `call_centre_id` was transformed to four variables `call_centre_id1`. . . `call_centre_id4`.

```
head(deb_cc_agg)
```

```
##   debtor_id call_centre_id1 call_centre_id2 call_centre_id3
## 1         1               1               0               0
## 2         2               0               1               0
## 3         4               0               0               1
## 4         6               1               0               0
## 5         8               0               0               0
## 6         9               0               0               0
##   call_centre_id4
## 1               0
## 2               0
## 3               0
## 4               0
## 5               1
## 6               1
```

Each of them shows a number of times the certain debtor was contacted by each call centre, so now data in `deb_cc_agg` are clean.

Then we merge `raw_df` and `deb_cc_agg` data frames by inner join to avoid N/A values for `debtor_id`s not covered by calls.

```
#### join raw_df and deb_cc_agg
final_df <- merge(raw_df, deb_cc_agg, by = 'debtor_id')
```

Now we are coming to a clean data set to build a model.

```
final_df$debtor_id <- NULL

#### creating a clean data set
my_cols <- !colnames(final_df) %in% c('output')
clean_df <- final_df[, my_cols]
clean_df$output <- final_df$output
```

As before, we are excluding highly correlated variables to avoid overfitting.

```r
num_cols <- !colnames(clean_df) %in% c('sex', 'marriage', 'education',
                                        'output')
corr_mtx <- cor(clean_df[, num_cols])
highly_correlated <- findCorrelation(corr_mtx, cutoff = 0.7)
hc_names <- colnames(corr_mtx)[highly_correlated]
clean_names <- setdiff(names(clean_df), hc_names)
clean_df <- clean_df[, clean_names]
```

And finally we use one-hot encoding to transform the rest of factor variables.

```r
# one-hot-encoding categorical variables
dummied_df <- dummy.data.frame(clean_df,
            names = c('sex', 'marriage', 'education'), sep="")
```

Building Models

Let's split the `dummied_df` data frame to training and testing subsets.

```r
set.seed(1234)
trainIndices <- createDataPartition(dummied_df$output,
                                    p = 0.7, list = FALSE)
train <- dummied_df[trainIndices, ]
test <- dummied_df[-trainIndices, ]

features <- colnames(train)[1:length(train)-1]
```

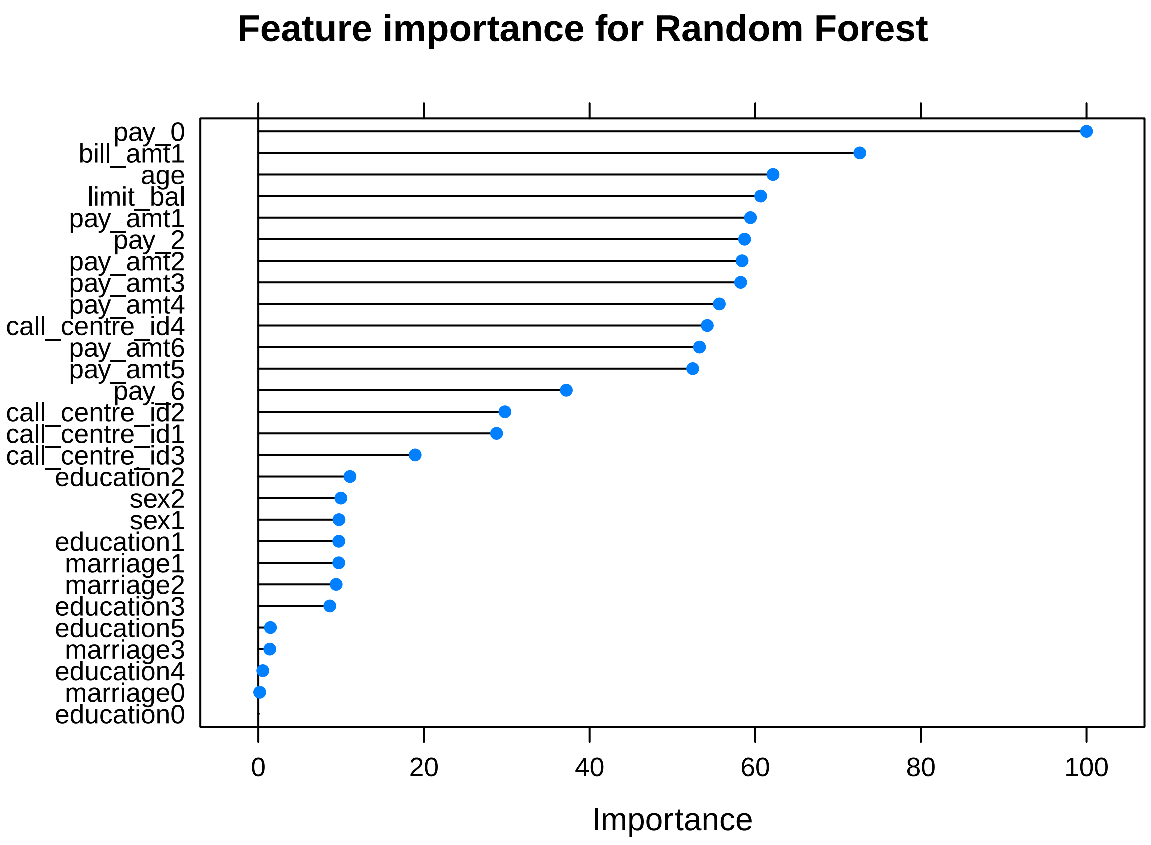
Again we use cross-validation as a resampling method.

```r
# resampling, k-fold cross-validation
fit_ctl <- trainControl(method = 'cv', number = 5)
```

We will fit only our two best models to get the idea on the possible improvement.

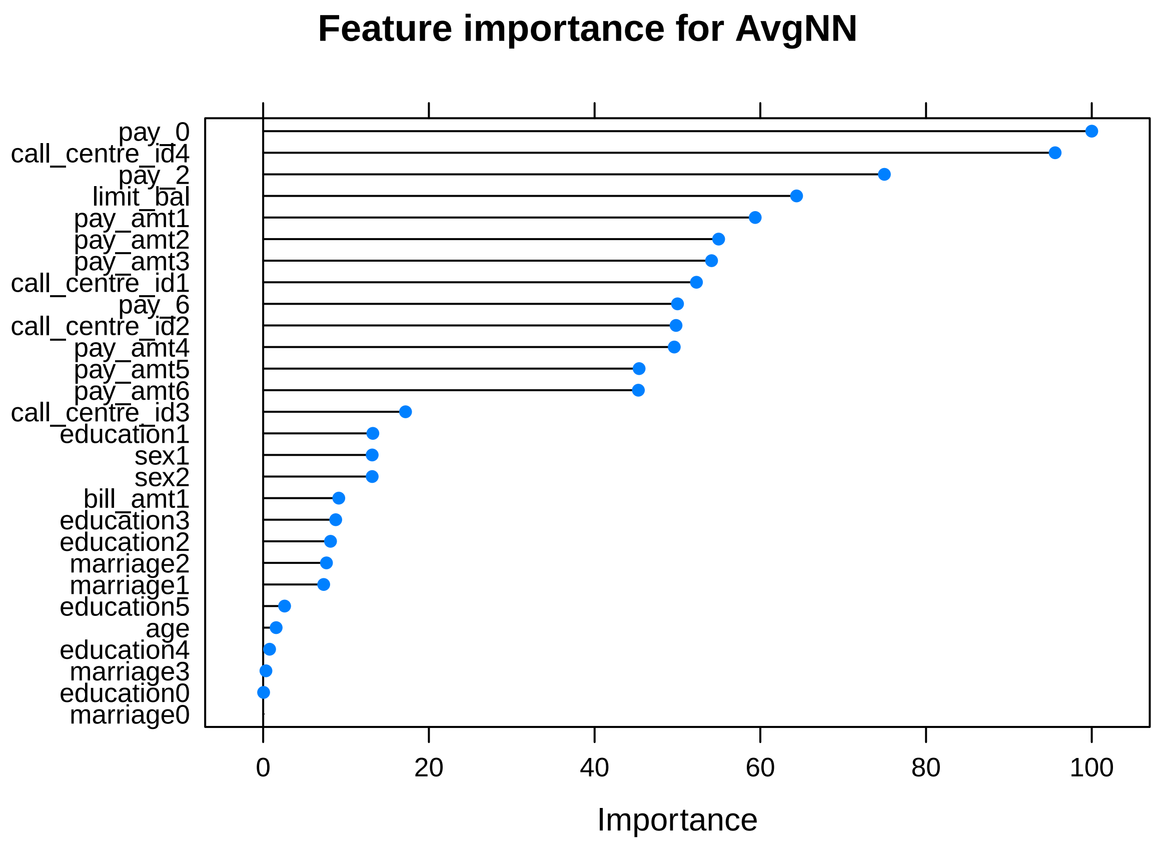
Random Forest

```r
#### Random Forest
fit_rf <- train(x = train[, features], y = train$output,
                method = 'rf',
                preProcess = 'range',
                trControl = fit_ctl,
                tuneGrid = expand.grid(.mtry = c(2:6)),
                n.tree = 500)
predict_rf <- predict(fit_rf, newdata = test[, features])
confMat_rf <- confusionMatrix(predict_rf, test$output, positive = '1')
importance_rf <- varImp(fit_rf, scale = TRUE)

plot(importance_rf, main = 'Feature importance for Random Forest')
```



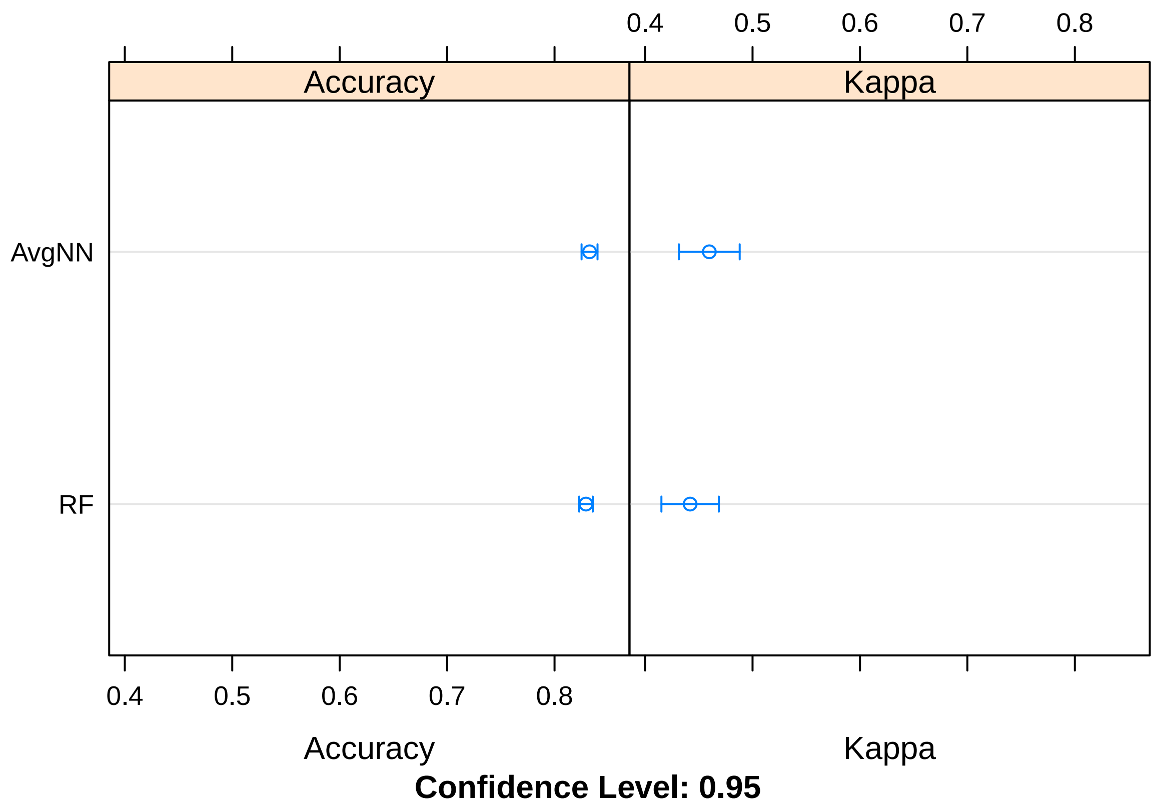
Averaged Neural Networks

```
# avNNet, ensemble method
fit_nn <- train(x = train[, features], y = train$output,
                    method = 'avNNet',
                    preProcess = 'range',
                    trControl = fit_ctl,
                    tuneGrid = expand.grid(.size = c(3, 5),
                                           .decay = c(0, 0.001, 0.01),
                                           .bag = FALSE),
                    trace = FALSE,
                    maxit = 1000)
predict_nn <- predict(fit_nn, newdata = test[, features])
confMat_nn <- confusionMatrix(predict_nn, test$output,
                                   positive = '1')
importance_nn <- varImp(fit_nn, scale = TRUE)

plot(importance_nn, main = 'Feature importance for AvgNN')
```



Results

```
models <- resamples(list(RF = fit_rf,
                         AvgNN = fit_nn
                         ))

dotplot(models)
```



As a result of using aditional data sets we were able to improve the old models by accuracy (RF: 0.8324122; AvgNN: 0.8377763) and expected out-of-sample error rate (Kappa: 0.4596797 and 0.4788866 respectively).

We can conclude that the information on the phone calls placed by agents is important to predict an outcome. The new models take into account the call centres IDs assuming that this kind of feature consolidates all the factors influencing the call centres performance and not fully represented in the currently available data sets. More than this, we kept the complexity of the models at almost the same level.

Acknowledgements

Lichman, M. (2013). (UCI Machine Learning Repository) [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.