Code


```
#### Introduction ####
# In the past, the best submissions for this data mining challenge
# (the challenge not the data set) have been simple logistic
# regression models that use variables that make intuitive sense. It is absolutely paramount for a data
# scientist to understand the domain the data is from. There are examples where 10 Phd guys build
# some ridiculous NN and then there is this guy who regresses three variables out of the 100 or so
# and comes up with a better model.
# Having said that, I am lazy and intrigued at the same time. I don't want to dig into the data
# too much, but just want to build a classifier for the hell of it. Classic mistake.
# There are two reasons why in this particular case this is a bad idea. First, the data is not
# very complex. Second, it is a very intuitive case. If these conditions are met, it is likely better
# to build a simple classifier, instead of going ham on the methods that are available, like
# building a random forest. This approach generally means you have no idea what you're dealing
# with and/or are just enjoying the academic challenge.
# This should be the lesson of this competition, KISS - Keep It Simple Stupid. Too bad it won't stick.
# So let's go down the rabbit hole.
require(Matrix)
require(data.table)

# read data
d <- read.csv('../input/carInsurance_train.csv')
# classify <- read.csv('~/Desktop/ML/DSS_DMC_2017/Working Material/test_set.csv')

##########################################################################################
#### data exploration ####
## should probably do outliers and data exploration
boxplot(d) # wow, balance. Could think about normalizing it and kicking the guy up top out. Let's check
# if balance correlates with the Car Insurance to be classified

cor.test(d$Balance, d$CarInsurance)
# pretty siginificant. What if we remove some "outliers"?
cor.test(d[d$Balance <= 20000,"Balance"], d$CarInsurance[d$Balance <= 20000])
# stops being significant. I am definitely overlooking some mathematical truth that would be apparent if I took the

## let's look at the others without balance throwing everything in disarray
boxplot(d[,-c(1,7)]) # obviously we don't need ID either
# days passed still messed up

cor.test(d$DaysPassed, d$CarInsurance)
# pretty relevant

#### We'll leave it at that. It should give you an idea. Caret is a good resource as well http://topepo.github.io/ca

##########################################################################################
#### Cleaning ####

### Cleaning has two parts. Getting rid of the usual, like handling NAs and dealing with nonsensical stuff (e.g. ID
## My assumptions are that the day and time of day for the calls influence conversion rates.
# ******* should probably look at the columns for LastCallDay and LastCallMonth. They don't make sense in their cur
# Actually, finding out which weekday calls were made to what time would be very helpful. Because cold calls tend t
# We start with the 2nd part first, because NA imputation also somewhat depends on the steps we will take here. You

## Get the weekdays in a workable format
# What are we doing here? We believe there is a chance that calls made at a certain time on certain days, e.g. Thur
# However, we don't know which year this data is from. Which does not matter. We just need to have a uniform distri

clean_d <- d

max(clean_d[clean_d$LastContactMonth == 'feb', 'LastContactDay']) # check if the year the calls were made isn't a l
```

```r
clean_d$DateCall <- as.Date(paste(clean_d$LastContactDay, clean_d$LastContactMonth, "2015", sep = '/'), "%d/%b/%Y")
clean_d$Weekday <- factor(weekdays(clean_d$DateCall))

# Next, we want to know what time people were called during the day.
# Let's see when the calls were made and what the working hours are.
plot(table(clean_d$CallStart)) # not very informative let's take the minutes and seconds off
plot(table(call_hr <- gsub("(:\\d{2})", "", clean_d$CallStart))) # ok... they are pretty diligent in calling people

# We could take the times as they are given. However, that would be too much noise, in my opinion. Therefore, I opt
clean_d$CallDayTime <- as.numeric(gsub("(:\\d{2})", "", clean_d$CallStart))
require(car)
clean_d$CallDayTime <- factor(recode(clean_d$CallDayTime, "c('9', '10', '11')='morning'; c('12', '13', '14')='midday


##
# Convert last phone call time to minutes
require(chron)
clean_d$call_dur_min <- 60 * 24 * as.numeric(times(clean_d$CallEnd)-times(clean_d$CallStart))

## find NAs
# I didn't come up with that. Google + Stackoverflow = Bliss
na_count <-sapply(clean_d, function(y) sum(length(which(is.na(y)))))
na_count <- data.frame(na_count)

## The issue is that we have a lot of NAs. Question is, what to do with them? Kick the missing values? Include them
# But most classifiers can't handle missing values. So we should replace them. For a more holistic
# overview of the how's and why's read the article above. We are going to go with the cool solution.
# We replace NAs with factors falling out of a k-Nearest-Neighbor

## Before we can do that, we need to take away all the columns that we won't consider in our analysis. Reason being
summary(clean_d)
# ID is the first to go, redundant a.f.. Outcome probably too much noise, has to go. CallStart and CallEnd, LastCont
# Kick everything we don't need. This is a little complicated in R if you want to do it by column name. We do it any

sub_clean_d <- subset(clean_d, select = -c(Id, LastContactDay, LastContactMonth, Outcome, CallStart, CallEnd, DateCa

# We can do the NA replacement now
## Impute NAs https://www.r-bloggers.com/missing-value-treatment/
# this might leave us in Hell's Kitchen. Well... actually it's Freedman's Kitchen https://www.r-bloggers.com/freedma

require(DMwR)
set.seed(42)

clean_d_imputeknn <- knnImputation(sub_clean_d)  # perform knn imputation.
# check if it missed any NAs
anyNA(clean_d_imputeknn)


########################################################################################
#### Building the actual model ####
## We start with a random forest.
## Using the caret package we cross validate before we train the model
# Cross validation is pretty much the most imortant thing, because it reduces overfitting. It means we partition the
# Basically with the caret package, we define a cross validation object which we pass to the actual function to crea

## Making a nice data frame for the modelling operations.
model_d <- clean_d_imputeknn


# Split data into test and train: so later we can validate our model
require(caret)
set.seed(42)

# We will use this index to define the rows, which will go into train and test data sets respectively
train_index <- createDataPartition(model_d$CarInsurance, p = 0.75, list = FALSE, times = 1)
training <- model_d[train_index, ]
testing <- model_d[-train_index, ]

# Cross validation - the train_control object will tell the model how to partition the data
train_control = trainControl(method="cv", number=10)


#### Random Forest ####
# Training the actual model. We have to pass CarInsurance, i.e. the variable to be classified, as a factor, not as a
set.seed(42)
model_rf = train(factor(CarInsurance)~., data=training, trControl=train_control, method="rf")

```

```
# We make a frame and fill it with the predicted values. This allows us to the test the quality of the model in the
prediction_rf = predict(model_rf, subset(testing, select=-c(CarInsurance)))

#Compute the accuracy of predictions with a confusion matrix
confusionMatrix(prediction_rf, testing$CarInsurance)

#### Logistic Regression ####
## simple first, then bagging and boosting
set.seed(42)
model_logreg <- glm(factor(CarInsurance) ~., family=binomial(link='logit'), data=training)

prediction_logreg = predict(model_logreg, subset(testing, select=-c(CarInsurance)), type='response') # by choosing

table(testing$CarInsurance, prediction_logreg > 0.5) # LogReg gives the results as probabilities, so we can't use th

## so Random Forest results are actually better. However, the RF takes about a minute or so to calculate the model.

#### LogitBoost ####
set.seed(42)
model_logitboost <- train(factor(CarInsurance)~., data=training, trainControl=train_control, method="LogitBoost", n

prediction_logitboost = predict(model_logitboost, subset(testing, select=-c(CarInsurance)))
confusionMatrix(prediction_logitboost, testing$CarInsurance)

#### XGB Trees ####
set.seed(42)
model_xgbtrees <- train(factor(CarInsurance)~., data=training, method='xgbTree', trainControl=train_control, metric=

prediction_xgbtrees = predict(model_xgbtrees, subset(testing, select= -c(CarInsurance)))
confusionMatrix(prediction_xgbtrees, testing$CarInsurance) # worst result yet :(
```

Did you find this Kernel useful?
Show your appreciation with an upvote

0

Run Info

Succeeded	False	Run Time	1200.5 seconds
Exit Code	137	Queue Time	0 seconds
Docker Image Name	kaggle/rstats (Dockerfile)	Output Size	0
Timeout Exceeded	True	Used All Space	False
Failure Message	The kernel was killed for running longer than 1200 seconds.		

Log

Download Log

```
Time   Line #  Log Message
1.3s        1  Loading required package: Matrix
2.2s        2  Loading required package: data.table
3.1s        3
3.1s        4          Pearson's product-moment correlation

               data:  d$Balance and d$CarInsurance
3.1s        5  t = 2.6302, df = 3998, p-value = 0.008567
               alternative hypothesis: true correlation is not equal to 0
               95 percent confidence interval:
                0.01058319 0.07245913
               sample estimates:
3.1s        6         cor
               0.04156101

                       Pearson's product-moment correlation

               data:  d[d$Balance <= 20000, "Balance"] and d$CarInsurance[d$Balance <= 20000]
3.1s        7  t = 4.2517, df = 3974, p-value = 2.17e-05
               alternative hypothesis: true correlation is not equal to 0
```

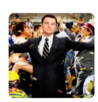
```
                 95 percent confidence interval:
                  0.03628313 0.09817159
                 sample estimates:
                        cor
                 0.06729209

3.2s        8
                         Pearson's product-moment correlation

                 data:  d$DaysPassed and d$CarInsurance
3.2s        9    t = 8.8714, df = 3998, p-value < 2.2e-16
                 alternative hypothesis: true correlation is not equal to 0
                 95 percent confidence interval:
                  0.1084183 0.1692057
                 sample estimates:
                       cor
                 0.1389429

                 [1] 27
3.3s       10    Loading required package: car
4.1s       11    Loading required package: chron
4.5s       12          Id             Age                   Job          Marital
                  Min.   :   1   Min.   :18.00   management :893   divorced: 483
                  1st Qu.:1001   1st Qu.:32.00   blue-collar:759   married :2304
                  Median :2000   Median :39.00   technician :660   single  :1213
                  Mean   :2000   Mean   :41.21   admin.     :459
                  3rd Qu.:3000   3rd Qu.:49.00   services   :330
4.5s       13     Max.   :4000   Max.   :95.00   (Other)    :880
                                                 NA's       : 19
                     Education       Default          Balance          HHInsurance
                  primary  : 561   Min.   :0.0000   Min.   :-3058.0   Min.   :0.0000
                  secondary:1988   1st Qu.:0.0000   1st Qu.:  111.0   1st Qu.:0.0000
                  tertiary :1282   Median :0.0000   Median :  551.5   Median :0.0000
                  NA's     : 169   Mean   :0.0145   Mean   : 1532.9   Mean   :0.4928
                                   3rd Qu.:0.0000   3rd Qu.: 1619.0   3rd Qu.:1.0000
                                   Max.   :1.0000   Max.   :98417.0   Max.   :1.0000

                    CarLoan        Communication  LastContactDay  LastContactMonth
                  Min.   :0.000   cellular :2831   Min.   : 1.00   may    :1049
                  1st Qu.:0.000   telephone: 267   1st Qu.: 8.00   jul    : 573
                  Median :0.000   NA's     : 902   Median :16.00   aug    : 536
                  Mean   :0.133                    Mean   :15.72   jun    : 454
                  3rd Qu.:0.000                    3rd Qu.:22.00   nov    : 347
                  Max.   :1.000                    Max.   :31.00   apr    : 306
                                                                   (Other): 735
                   NoOfContacts      DaysPassed      PrevAttempts        Outcome
                  Min.   : 1.000   Min.   : -1.00   Min.   : 0.0000   failure: 437
                  1st Qu.: 1.000   1st Qu.: -1.00   1st Qu.: 0.0000   other  : 195
                  Median : 2.000   Median : -1.00   Median : 0.0000   success: 326
                  Mean   : 2.607   Mean   : 48.71   Mean   : 0.7175   NA's   :3042
                  3rd Qu.: 3.000   3rd Qu.: -1.00   3rd Qu.: 0.0000
                  Max.   :43.000   Max.   :854.00   Max.   :58.0000

                    CallStart        CallEnd       CarInsurance      DateCall
                  10:42:44:   3   10:22:30:   3   Min.   :0.000   Min.   :2015-01-08
                  11:48:25:   3   10:52:24:   3   1st Qu.:0.000   1st Qu.:2015-05-08
                  13:54:34:   3   11:27:46:   3   Median :0.000   Median :2015-06-05
                  15:27:56:   3   09:04:02:   2   Mean   :0.401   Mean   :2015-06-21
                  15:48:27:   3   09:06:42:   2   3rd Qu.:1.000   3rd Qu.:2015-08-11
                  17:02:39:   3   09:12:47:   2   Max.   :1.000   Max.   :2015-12-30
                  (Other) :3982   (Other) :3985
                       Weekday       CallDayTime     call_dur_min
                  Friday   :725   afternoon:1351   Min.   : 0.08333
                  Monday   :454   midday   :1365   1st Qu.: 2.10000
                  Saturday :380   morning  :1284   Median : 3.86667
                  Sunday   : 92                    Mean   : 5.84740
                  Thursday :849                    3rd Qu.: 7.66667
                  Tuesday  :724                    Max.   :54.21667
                  Wednesday:776
4.5s       14    Loading required package: DMwR
4.6s       15    Loading required package: methods
4.7s       16    Loading required package: lattice
4.7s       17    Loading required package: grid
45.8s      18    [1] FALSE
45.8s      19    Loading required package: caret
45.8s      20    Loading required package: ggplot2
46.2s      21    Loading required package: randomForest
46.2s      22    randomForest 4.6-12
                 Type rfNews() to see new features/changes/bug fixes.
46.3s      23
                 Attaching package: 'randomForest'

                 The following object is masked from 'package:ggplot2':

                     margin

197.6s     24    Confusion Matrix and Statistics

                           Reference
                 Prediction   0   1
                          0 483  79
                          1 106 332
197.6s     25

                               Accuracy : 0.815
                                 95% CI : (0.7895, 0.8386)
                     No Information Rate : 0.589
                     P-Value [Acc > NIR] : < 2e-16
```

```
                  Kappa : 0.6216
 Mcnemar's Test P-Value : 0.05593

            Sensitivity : 0.8200
            Specificity : 0.8078
         Pos Pred Value : 0.8594
         Neg Pred Value : 0.7580
             Prevalence : 0.5890
         Detection Rate : 0.4830
   Detection Prevalence : 0.5620
      Balanced Accuracy : 0.8139

       'Positive' Class : 0

197.7s    26
              FALSE TRUE
            0   499   90
            1   117  294
198.3s    27  Loading required package: caTools
211.5s    28  Confusion Matrix and Statistics

                        Reference
              Prediction   0   1
                       0 506 147
                       1  83 264
211.5s    29
                     Accuracy : 0.77
                       95% CI : (0.7426, 0.7958)
          No Information Rate : 0.589
          P-Value [Acc > NIR] : < 2.2e-16

                        Kappa : 0.5135
       Mcnemar's Test P-Value : 3.266e-05

                  Sensitivity : 0.8591
                  Specificity : 0.6423
               Pos Pred Value : 0.7749
               Neg Pred Value : 0.7608
                   Prevalence : 0.5890
               Detection Rate : 0.5060
         Detection Prevalence : 0.6530
            Balanced Accuracy : 0.7507

             'Positive' Class : 0

211.6s    30  Loading required package: xgboost
211.6s    31  Loading required package: plyr
211.6s    32
              Attaching package: 'plyr'

              The following object is masked from 'package:DMwR':

                  join

211.6s    33
211.6s    35  Failed. Exited with code 137.
```

Data

Data Sources

- Car Insurance Cold Calls
  - carInsurance_test.csv — 19 columns
  - carInsurance_train.csv — 19 columns
  - DSS_DMC_Description.pdf



Car Insurance Cold Calls

We help the guys and girls at the front to get out of Cold Call Hell

Last Updated: 2 years ago (Version 1)

About this Dataset

Introduction

Here you find a very simple, beginner-friendly data set. No sparse matrices, no fancy tools needed to understand what's going on. Just a couple of rows and columns. Super simple stuff. As explained below, this data set is used for a competition. As it turns out, this competition tends to reveal a common truth in data science: KISS - Keep It Simple Stupid

What is so special about this data set is, given it's simplicity, it pays off to use "simple" classifiers as well. This year's competition was won by a C5.0 . Can you do better?

Description

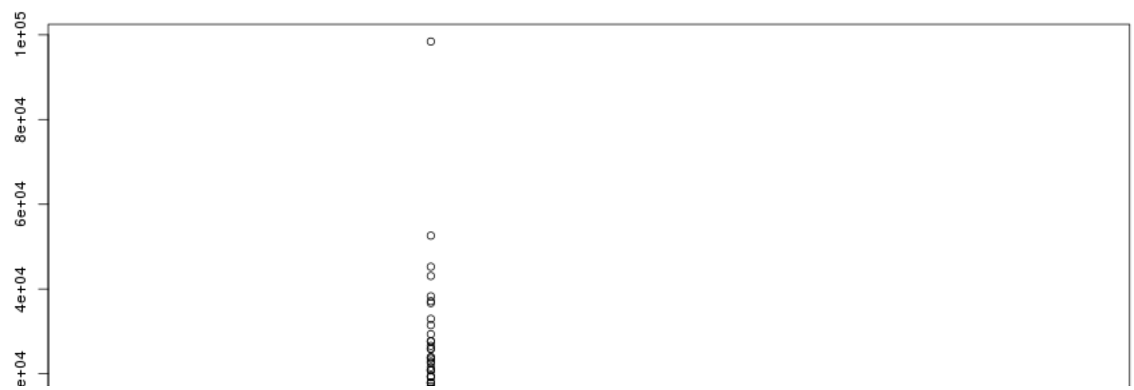
We are looking at cold call results. Turns out, same salespeople called existing insurance customers up and tried to sell car insurance. What you have are details about the called customers. Their age, job, marital status, whether the have home insurance, a car loan, etc. As I said, super simple.

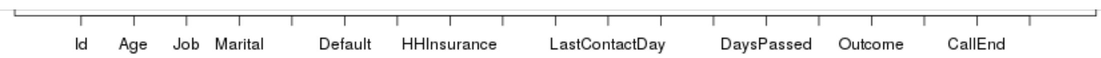
What I would love to see is some of you applying some crazy XGBoost classifiers, which we can square off against some logistic regressions. It would be curious to see what comes out on top. Thank you for your time, I hope you enjoy using the data set.

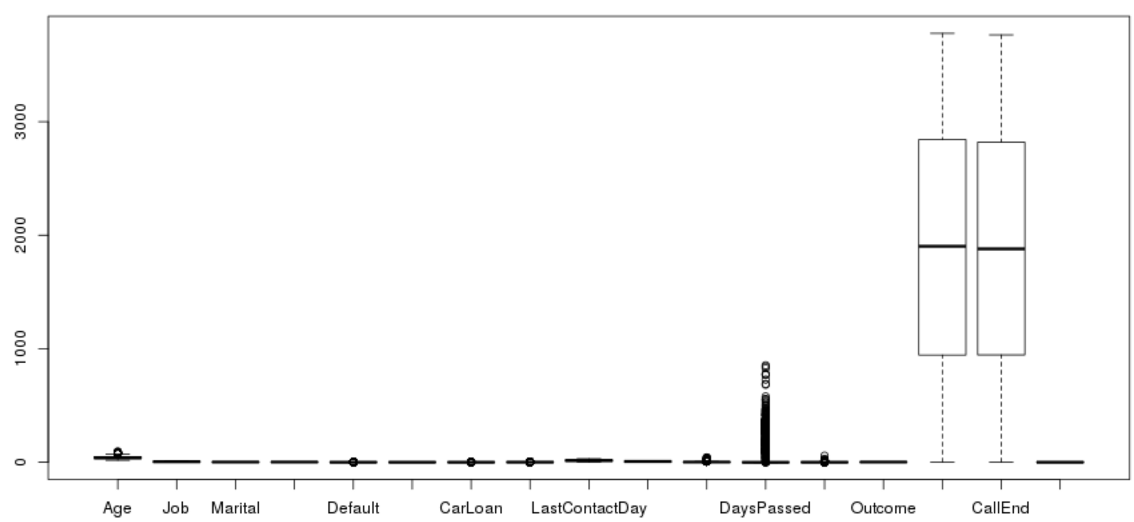
Acknowledgements

Thanks goes to the Decision Science and Systems Chair of

Output Visualizations







Output visualization 3

Comments (0)

Please sign in to leave a comment.

Code Log Data Output Comments

Our Team Terms Privacy Contact/Support