Detecting Fraudulent Activities

Siddhartha Jetti

December 02, 2016

1. Problem Defnition

E-commerce websites deal with a high risk of users performing fraudulent activities. Machine Learning really excels at identifying fraudulent activities.The goal of this project is to build a machine learning model that predicts the probability that the first transaction of a new user is fraudulent.

a. Load libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.3.3
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.3.3
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

b. Read datasets

```
root_directory <- "C:\\Users\\ub71493\\Desktop\\Fraudulant Activities"
setwd(root_directory)
data = read.csv("./Input/Fraud_Data.csv",stringsAsFactors = F)
ip_addresses = read.csv("./Input/IpAddress_to_Country.csv",stringsAsFactors = F)
```

2. Process Data

Check existence of duplicates

```
nrow(data) == length(unique(data$user_id))
```

```
## [1] TRUE
```

Add country to original dataset based on ip addresses

```
data_country = rep(NA, nrow(data))
for (i in 1: nrow(data)){
tmp = as.character(ip_addresses [data$ip_address[i] >= ip_addresses$lower_bound_ip_address & data$ip_address[i] <= ip_addres
ses$upper_bound_ip_address,"country"])
if (length(tmp) == 1) {data_country[i] = tmp}
}
data$country = data_country
data[, "signup_time"] = as.POSIXct(data[, "signup_time"], tz="GMT")
data[, "purchase_time"] = as.POSIXct(data[, "purchase_time"], tz="GMT")
summary(as.factor(data$country))
```

```
##                   United States                           China
##                           58049                           12038
##                           Japan                  United Kingdom
##                            7306                            4490
##               Korea Republic of                         Germany
##                            4162                            3646
##                          France                          Canada
##                            3161                            2975
##                          Brazil                           Italy
##                            2961                            1944
##                       Australia                     Netherlands
##                            1844                            1680
##              Russian Federation                           India
##                            1616                            1310
## Taiwan; Republic of China (ROC)                          Mexico
##                            1237                            1121
##                          Sweden                           Spain
##                            1090                            1027
##                    South Africa                     Switzerland
##                             838                             785
##                          Poland                       Argentina
##                             729                             661
##                       Indonesia                          Norway
##                             649                             609
##                        Colombia                          Turkey
##                             602                             568
##                        Viet Nam                         Romania
##                             550                             525
##                         Denmark                       Hong Kong
##                             490                             471
##                         Finland                         Austria
##                             460                             435
##                         Ukraine                           Chile
##                             429                             417
##                         Belgium      Iran (ISLAMIC Republic Of)
##                             409                             389
##                           Egypt                  Czech Republic
##                             359                             349
##                        Thailand                     New Zealand
##                             291                             278
```

```
##                         Israel                   Saudi Arabia
##                            272                            264
##                      Venezuela                        Ireland
##                            251                            240
##                 European Union                         Greece
##                            238                            231
##                       Portugal                        Hungary
##                            229                            211
##                       Malaysia                      Singapore
##                            210                            208
##                       Pakistan                    Philippines
##                            186                            177
##                       Bulgaria                        Morocco
##                            166                            158
##                        Algeria                           Peru
##                            122                            119
##                        Tunisia            United Arab Emirates
##                            118                            114
##                        Ecuador                      Lithuania
##                            106                             95
##                     Seychelles                          Kenya
##                             95                             93
##                     Kazakhstan                     Costa Rica
##                             92                             90
##                         Kuwait                       Slovenia
##                             90                             87
##      Slovakia (SLOVAK Republic)                        Uruguay
##                             86                             80
##  Croatia (LOCAL Name: Hrvatska)                        Belarus
##                             79                             72
##                     Luxembourg                         Serbia
##                             72                             69
##                        Nigeria                         Latvia
##                             67                             64
##                         Panama                        Bolivia
##                             62                             53
##              Dominican Republic                         Cyprus
##                             51                             43
##                        Estonia                           Oman
##                             42                             41
##                     Bangladesh             Moldova Republic of
```

```
##                          37                          37
##                    Paraguay                     Georgia
##                          35                          32
##                   Sri Lanka       Bosnia and Herzegowina
##                          31                          30
##                 Puerto Rico                      Jordan
##                          30                          28
##                     Lebanon                 El Salvador
##                          28                          25
##                       Qatar                       Sudan
##                          25                          25
##                      Angola                    Macedonia
##                          24                          24
##          Syrian Arab Republic                  Azerbaijan
##                          24                          23
##                     Namibia                      Malta
##                          23                          22
##                     (Other)                      NA's
##                         550                       21966
```

3 Feature Engineering

A few obvious variables that can be created here could be: 1. Time difference between sign-up time and purchase time 2. If the device id is unique or certain users are sharing the same device (many different user ids using the same device could be an indicator of fake accounts). 3. Many different users having the same ip address could be an indicator of fake accounts. 4. Usual week of the year and day of the week from time variables.

Compute time difference between purchase and signup

```
data$purchase_signup_diff = as.numeric(difftime(as.POSIXct(data$purchase_time, tz="GMT"), as.POSIXct(data$signup_time, tz="G
MT"), unit="secs"))
```

check for each device id/ip address how many different users had it

```
data <- data %>%
  group_by(device_id) %>%
  mutate (device_id_count = n())

data <- data %>%
  group_by(ip_address) %>%
  mutate (ip_address_count = n())
```

Day of the week and week of the year

```
data$signup_time_wd = format(data$signup_time, "%A")
data$purchase_time_wd = format(data$purchase_time, "%A" )
data$signup_time_wy = as.numeric(format(data$signup_time, "%U"))
data$purchase_time_wy = as.numeric(format(data$purchase_time, "%U" ))
```

Drop unwanted variables from dataset

```
data_rf = data[, -c(1:3, 5)]
```

keep the top 50 countries and everything else is “other”

```
data_rf$country[is.na(data_rf$country)]="Not_found"
data_rf$country = ifelse(data_rf$country %in% names(sort(table(data_rf$country),decreasing = TRUE))
[51:length(unique(data_rf$country))],"Other", as.character(data_rf$country))
```

Process required variables

```
#make class a factor
data_rf$class = as.factor(data_rf$class)
#all characters become factors
data_rf[sapply(data_rf, is.character)] <- lapply(data_rf[sapply(data_rf, is.character)], as.factor)
```

4. Train models and score test dataset

The model is fit using 2/3 of data as training and remaining 1/3 of the data as test dataset.

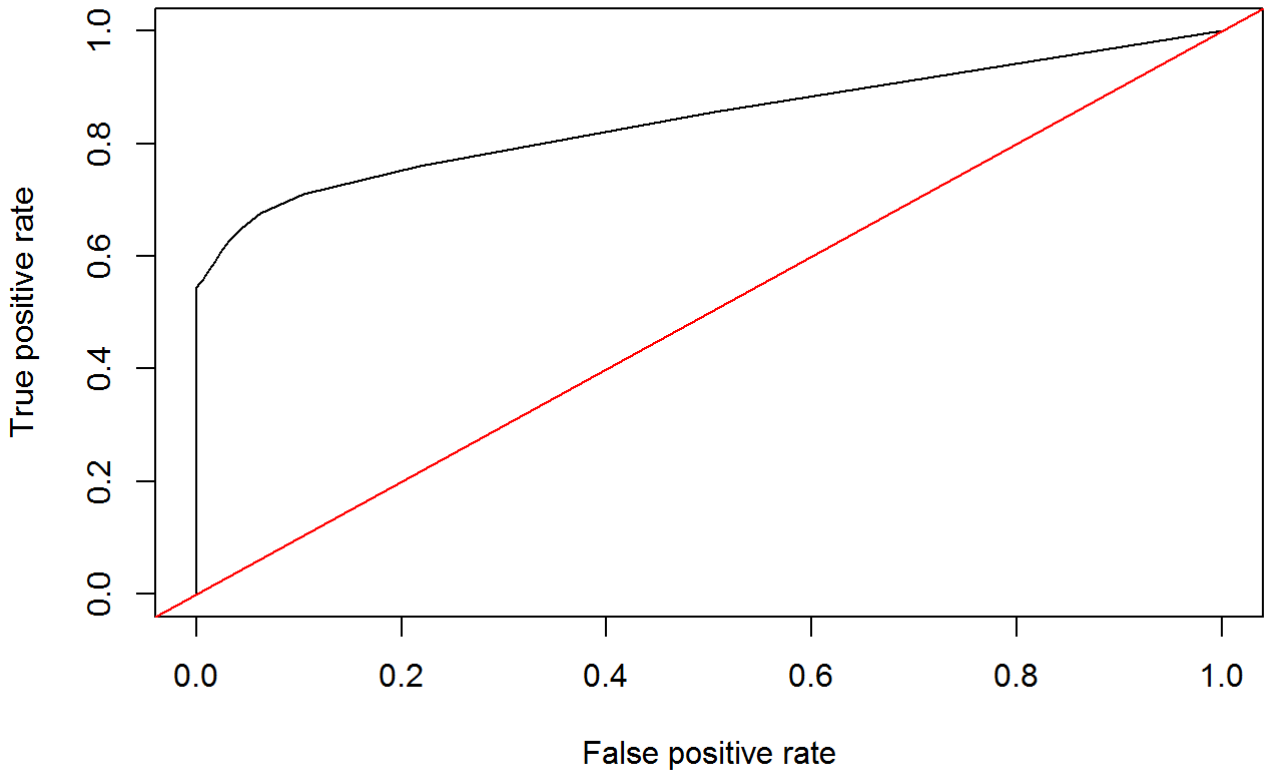
```
train_sample = sample(nrow(data_rf), size = nrow(data)*0.66)
train_data = data_rf[train_sample,]
test_data = data_rf[-train_sample,]
```

For the given problem lets choose random forest method for prediction. The random forests usually require very little time to optimize it (its default params are often close to the best ones) and are robust with outliers, irrelevant variables, continuous and discrete variables. Also they provide partial dependence plots and variable importance to get insights about how information is derived from the variables. The Random forest method is known to be flexible and has relatively lower variance than an individual tree.

```
rf = randomForest(y=train_data$class, x = train_data[, -7],ytest = test_data$class,
                  xtest = test_data[, -7],ntree = 50, mtry = 3, keep.forest = TRUE)
print(rf)
```

```
##
## Call:
##  randomForest(x = train_data[, -7], y = train_data$class, xtest = test_data[,      -7], ytest = test_data$class, ntree =
 50, mtry = 3, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 4.39%
## Confusion matrix:
##       0    1  class.error
## 0 90343   12 0.0001328095
## 1  4365 5013 0.4654510557
##                 Test set error rate: 4.25%
## Confusion matrix:
##       0    1  class.error
## 0 46602    4 8.582586e-05
## 1  2181 2592 4.569453e-01
```

```
#let's combine in one data set model predictions and actual values.
#The first column are the actual classes in our test set and the second are the predicted scores
rf_results = data.frame (true_values = test_data$class,predictions = rf$test$votes[,2])
pred = prediction (rf_results$predictions, rf_results$true_values)
#plot the ROC and look at true positive vs false positive
perf = performance (pred, measure = 'tpr', x.measure = "fpr")
plot(perf) + abline(a=0, b=1, col = 'red') # the red line is randomness
```




```
## numeric(0)
```

5. Conclusion.

The model produces the probability of a new user committing fraud and a suitable cut-off probability must be selected to classify a particular user as fraud.By default random forest method uses 0.5 as cutoff.If priority is to minimize false positive, a cut-off that gives true positive rate of ~0.5 and false positive rate almost zero (this was essentially the random forest output) should be chosen. However, if we care about maximizing true positive, we will have to decrease the cut-off. This way we will classify more events as “1”: some will be true ones (so true positive goes up) and many unfortunately, will be false ones (so false positive will also go up).